TESIS – KI142502

DETEKSI KONFLIK LEKSIKAL PADA DIAGRAM KELAS MENGGUNAKAN MODIFIKASI GRAF DAN SIMILARITAS WORDNET

Billy Montolalu
5111201026

DOSEN PEMBIMBING
Dr. Ir. Siti Rochimah, MT.
NIP. 196810021994032001

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

THESIS – KI142502

CLASS DIAGRAM LEXICON CONFLICT DETECTION USING GRAPH MODIFICATION AND WORDNET SIMILARITY

Billy Montolalu
5111201026

SUPERVISOR
Dr. Ir. Siti Rochimah, MT.
NIP. 196810021994032001

MASTER PROGRAM
SOFTWARE ENGINEERING
DEPARTEMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Komputer (M.Kom.)

di

Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

BILLY MONTOLALU

NRP. 5111201026

Dengan Judul :

Deteksi Konflik Leksikal Pada Diagram Kelas Menggunakan Modifikasi Graf Dan Similaritas WordNet

Tanggal Ujian : 10-01-2017

Periode Wisuda : 2016 Gasal

Disetujui oleh :

1. Dr. Ir. Siti Rochimah, MT
   NIP. 196810021994032001 (Pembimbing)
2. Daniel Oranova, S,Kom. MSc. PD.Eng.
   NIP. 197411232006041001 (Penguji)
3. Fajar Baskoro, S.Kom, M.T
   NIP. 197404031999031002 (Penguji)
4. Rizky Januar Akbar, S.Kom., M.Eng.
   NIP. 198701032014041001 (Penguji)

an. Direktur Program Pascasarjana
Asisten Direktur

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 196[illegible]21 198603 1 001

Direktur Program Pasca Sarjana,

Prof.Ir.Djauhar Manfaat, M.Sc., Ph.D.
NIP. 19601202198701100

[Halaman ini sengaja dikosongkan]

DETEKSI KONFLIK LEKSIKAL PADA DIAGRAM KELAS MENGGUNAKAN MODIFIKASI GRAF DAN SIMILARITAS WORDNET

Nama mahasiswa : Billy Montolalu
NRP : 5110201026
Pembimbing : Dr. Ir. Siti Rochimah, MT.

ABSTRAK

Pada lingkungan pengembangan perangkat lunak banyak pengguna, konflik dapat terjadi karena pengguna melakukan perubahan pada bagian yang sama. Lingkungan pengembangan banyak pengguna menggunakan kontrol versi untuk mengelolah perubahan yang terjadi. Ketika salah satu pengguna melakukan penyimpanan, kontrol versi mendeteksi terjadinya konflik. Pendeteksian konflik dilakukan untuk mencegah kloning dan kesalahan sintaksis pada proses penggabungan dua buah versi.

Pada penelitian ini digunakan sebuah metode untuk mendeteksi konflik secara leksikal pada diagram kelas dalam lingkungan pengembangan banyak pengguna. Metode ini menggunakan pendekatan modifikasi graf dan similaritas WordNet. Pendekatan modifikasi graf digunakan untuk mencari bagian dalam diagram kelas yang hanya mengalami perubahan. Kelas, atribut, operasi dan relasi dimodelkan menjadi bentuk graf. Pendekatan ini menghasilkan graf dengan aturan minimal. Graf dengan aturan minimal adalah graf yang hanya mengalami perubahan. Konflik dapat dideteksi dengan membandingkan similaritas WordNet dua buah versi graf dengan aturan minimal. Pasangan bagian yang mempunyai nilai similaritas tinggi teridentifikasi sebagai kasus konflik leksikal.

Dalam penelitian ini metode diujikan dengan menggunakan diagram kelas yang terdiri dari 30 versi. Dari penelitian ini pendekatan yang diambil diperoleh nilai kappa sebesar 0,6404 yang menunjukkan bahwa hasil tingkat kecocokan antara sistem dan pakar pada penelitian ini adalah kuat (good).

Kata Kunci: Kontrol Versi Model, Deteksi Konflik, Konflik Leksikal

[Halaman ini sengaja dikosongkan]

CLASS DIAGRAM LEXICON CONFLICT DETECTION USING GRAPH MODIFICATION AND WORDNET SIMILARITY

Name	: Billy Montolalu
Student Identity Number	: 5111201026
Supervisor	: Dr. Ir. Siti Rochimah, MT.

ABSTRACT

In multi-user software development environtment, conflict occurs when more than one user make a change in the same part of the program. Multi-user software development environtment uses version control to manage every changes. When one user make a change, version control check for conflict. The Conflict Checking is usefull for preventing cloning and sintax error in multi-version development process.

In this research, a method is proposed for detecting conflict by checking class diagram from multi-user development environtment lexically. The proposed method uses modified graf and WordNet Similarity approach. The modified graf is used for searching the changed part in class diagram. Class, attribute, operation, and relation are modeled as a graf. This approach produces a graf with minimum rule. A minimum rule graf is a graf that contains only the changes. Conflict can be detected by comparing WordNet similarization and 2 version of the graf with minimum rule. The one with higer similarity value is detected as lexical conflict case.

The proposed method is tested using class diagram which has more than 30 version. The result shows that the method obtains the kappa value 0.6404. This value indicates that the compatibiltiy between the method and expert is strong.

Key Words: Model Version Control, Conflict Detection, Lexicon Conflict

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah yang telah memberikan rahmatnya sehingga penulis dapat menyelesaikan tesis dengan judul "Deteksi Konflik Leksikal Pada Diagram Kelas Menggunakan Modifikasi Graf dan Similaritas WordNet" yang telah tertunda cukup lama. Tesis ini disusun untuk memenuhi sebagian persyaratan guna memperoleh gelar Magister Komputer di Institut Teknologi Sepuluh Nopember.

Melalui pengantar ini penulis ingin mengucapkan banyak terima kasih karena dalam penyusunan tesis ini penulis telah mendapat bantuan dan dorongan baik secara moril maupun materiil dari berbagai pihak. Untuk itu pada kesempatan ini penulis ingin mengucap terima kasih kepada :

1. Bapak Waskitho Wibisono, S.Kom, M.Eng, Ph. D, selaku Ketua Program Studi Pascasarjana Teknik Informatika Institut Teknologi Sepuluh Nopember.
2. Ibu Dr. Ir. Siti Rochimah, MT, selaku pembimbing dan penasehat akademik yang telah dengan sabar dan luar biasa membimbing dan mengarahkan penulis dalam pengerjaan tesis ini sehingga dapat terselesaikan dengan baik.
3. Keluarga penulis, Ibu, Bapak, Istriku Desy Nur Kumalasari dan anakku Kaisa yang selalu memberikan dukungan mental dan spiritual selama penulis menyelesaikan studinya di Institut Teknologi Sepuluh Nopember.
4. Teman-teman di Digital Medika yang telah membantu pekerjaan penulis.
5. Teman-teman Pascasarjana Teknik Informatika ITS, yang telah membantu dan berbagi informasi dengan penulis.
6. Teman-teman yang sudah membantu dalam mengisi angket studi kasus.

Serta semua pihak yang tidak dapat penulis sebutkan yang telah membantu penulis dalam penyelesaian tesis ini.

Penulis sangat menyadari bahwa tesis ini masih banyak kekurangan dan masih sangat mungkin untuk dikembangkan menjadi lebih baik, oleh karena itu penulis mengharapkan masukan dari semua pihak.

Surabaya, 19 November 2016

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN TESIS i
ABSTRAK iii
ABSTRACT v
KATA PENGANTAR vii
DAFTAR ISI ix
DAFTAR GAMBAR xi
DAFTAR TABEL xiii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan dan Manfaat Penelitian 3
BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI 5
2.1 Deteksi Konflik Pada Model 5
2.2 Kategorisasi Konflik 5
2.3 Version Control System (VCS) 11
2.4 Modifikasi Graf 12
2.5 Transformasi Graf 13
2.6 Karakteristik konflik 15
2.7 Similaritas WordNet 15
2.7.1. Similaritas antar kelas 16
2.7.2. Menghitung Similaritas WordNet 17
2.8 Konvensi Kode Java 19

2.9 Koefisien Cohen’s Kappa 21
BAB 3 METODE PENELITIAN 23
3.1 Studi Literatur 23
3.2 Perancangan Sistem 23
3.3 Pengembangan Perangkat Lunak 31
3.4 Pengujian dan Analisis 31
3.5 Penulisan Buku Tesis 32
BAB 4 HASIL DAN PEMBAHASAN 35
4.1 Pengumpulan Dataset 35
4.2 Implementasi 36
4.2.1. Pengolahan Dataset 37
4.2.2. Pembangun Graf 39
4.2.3. Pendeteksian Konflik Dengan Graf 39
4.2.4. Pendeteksian Konflik Dengan Similaritas WordNet 40
4.3 Skenario Pengujian 41
4.3.1. Pengujian 41
4.3.2. Analisis hasil pengujian 42
BAB 5 KESIMPULAN DAN SARAN 45
5.1 Kesimpulan 45
5.2 Saran 45
DAFTAR PUSTAKA 47
BIOGRAFI PENULIS 49

DAFTAR GAMBAR

Gambar 2.1 Konflik Tekstual 7
Gambar 2.2 Konflik Sintaksis 8
Gambar 2.3 Konflik Semantik Kontradiksi Statik 9
Gambar 2.4 Konflik Semantik Kontradiksi Perilaku 10
Gambar 2.5 Konflik Semantik Kontradiksi Arti 11
Gambar 2.6 Konflik pada VCS 12
Gambar 2.7 Modifikasi Graf (Taentzer, 2010) 12
Gambar 2.8 Transformasi Graf (Taentzer, 2010) 13
Gambar 2.9 Inisial pushout 14
Gambar 2.10 Kontruksi Aturan Minimal 14
Gambar 2.11 Aturan Minimal $PA = (LA \leftarrow KA \rightarrow RA)$ 15
Gambar 3.1 Metode pendeteksi konflik 24
Gambar 3.2 Diagram Kelas versi 0 (v_0) 24
Gambar 3.3 Diagram Kelas hasil modifikasi 25
Gambar 3.4 Hasil transformasi kedalam bentuk graf 26
Gambar 3.5 Graf dengan aturan minimal 27
Gambar 3.6 Hasil penggabungan v_1 dan v_2 28
Gambar 3.7 Hasil akhir penggabungan v_1 dan v_2 29
Gambar 3.8 Model Graf 30
Gambar 3.9 Graf v_0 dan v_1 dengan aturan minimal 30
Gambar 3.10 Graf v_0 dan v_2 dengan aturan minimal 31
Gambar 4.1 Diagram Kelas Sistem Informasi Parkiran Mobil 36
Gambar 4.2 Alur Kerja Pendeteksian Konflik Secara Otomatis 37
Gambar 4.3 Format Penyimpanan Diagram Kelas 38
Gambar 4.4 Diagram Kelas dari Objek-objek Penyusun Graf 39
Gambar 4.5 Antarmuka Hasil Deteksi Konflik 40

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 2.1 Penelitian Terkait Pendeteksian Konflik Pada Model..... 5
Tabel 2.2 Matrik Kemiripan Diagram Kelas M1 dan M2 18
Tabel 2.3 Matrik Kemiripan Dengan Nilai Tertinggi..... 18
Tabel 2.4 Matrik Kemiripan Tertinggi Iterasi Kedua 19
Tabel 2.5 Matrik Kemiripan Tertinggi Iterasi Terakhir..... 19
Tabel 2.6 Konvensi Penamaan Pada Java 19
Tabel 2.7 Tabel Relevansi..... 21
Tabel 2.8 Interpretasi Nilai Kappa (Altman, 1991)..... 22
Tabel 4.1 Hasil Deteksi Konflik oleh Pakar..... 42
Tabel 4.2 Perbandingan Hasil Deteksi Sistem 42
Tabel 4.3 Perhitungan Koefisien Kappa 43

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada industri rekayasa kebutuhan artefak perangkat lunak seperti kode sumber, dokumentasi, dan artefak lainnya disimpan kedalam suatu repositori. Repositori ini bisa diakses oleh banyak pengguna dan menyimpan semua perubahan yang ada pada artefak. Artefak kode sumber biasanya disimpan dalam suatu repositori khusus yang dinamakan dengan VCS (*Version Control System*). Dengan adanya VCS ini semua perubahan yang ada didalam kode sumber akan disimpan kedalam suatu versi-versi dan ada manajemen konflik jika ada pengguna melakukan perubahan di kode sumber yang sama. Dengan adanya VCS ini pengguna bisa berkolaborasi dan bekerja secara tim untuk membuat suatu perangkat lunak.

Sama halnya seperti kode sumber, artefak model seperti diagram pengguna, diagram kelas dan diagram aktifitas dibangun oleh banyak pengguna dan sering mengalami perubahan. Oleh karena itu manajemen konflik untuk artefak model juga diperlukan dalam pengembangan perangkat lunak.

VCS dalam kode sumber menggunakan perbandingan berbasis teks untuk manajemen versi dan konflik. Meskipun artefak model disimpan kedalam teks, pendekatan yang digunakan pada manajemen konflik VCS untuk kode sumber tidak akan sesuai jika diterapkan untuk artefak model. Hal ini dikarenakan seorang pemrogram akan lebih familiar dengan sintaksis berbasis teks dibandingkan dengan seorang pemodel yang terbiasa dengan sintaksis berbasis grafis (Brosch dkk, 2011). Disamping itu jika VCS pada kode sumber diterapkan untuk model, struktur graf pada model akan rusak dan informasi sintaksis dan semantik akan hilang (Taentzer dkk, 2010). Oleh karena itu perlu pendekatan khusus untuk menangani manajemen konflik pada model.

Beberapa penelitian telah dilakukan terkait dengan deteksi konflik pada model. Penelitian yang dilakukan oleh Brosch (Broch dkk, 2011) adalah membuat bahasa pemodelan untuk visualisasi konflik dengan menggunakan *UML Profile*.

Pada penelitian tersebut konflik dimodelkan secara visual dan visualisasi ini digunakan untuk memandu modeler melakukan penyelesaian terhadap konflik.

Penelitian selanjutnya dilakukan oleh Taentzer (Taentzer dkk, 2010) adalah melakukan pendeteksian terhadap konflik dengan menggunakan modifikasi graf. Pada penelitian tersebut dua buah model yang dibandingkan dimodelkan kedalam suatu graf. Graf tersebut mewakili perubahan yang terjadi pada setiap model. Kemudian pendeteksian konflik dilakukan dengan membandingkan graf kedua model berdasarkan tahap perubahan.

Rajbhoj (Asha Rajbhoj dan Sreedhar Reddy, 2013) melakukan penelitian yang sama dengan menggunakan graf. Penelitian ini dilakukan dengan cara menganalisa metamodel dari model yang dibandingkan. Sehingga pendeteksian konflik bisa digunakan untuk berbagai macam jenis model.

Ehrig (Ehrig dkk, 2011) melakukan penelitian untuk mencari resolusi terjadinya konflik dengan menggunakan modifikasi graf. Dalam penelitian ini metode modifikasi graf digunakan untuk mendeteksi konflik dan melakukan resolusi (menggabungkan dua buah graf yang saling konflik).

Penelitian dengan menggunakan metode modifikasi graf digunakan untuk menyelesaikan konflik secara sintaksis, struktural dan semantik. Akan tetapi penyelesaian konflik secara semantik dengan menggunakan modifikasi graf hanya didasarkan pada persamaan prilaku disetiap tahap perubahan model. Sehingga penerapan metode modifikasi graf ini tidak bisa digunakan untuk menyelesaikan konflik semantik secara bahasa.

Penyelesaian konflik yang tidak bisa diselesaikan secara otomatis harus diselesaikan dengan mempertemukan dua belah pihak yang melakukan konflik. Brosch (Brosch dkk, 2009) membuat suatu media yang bisa mempertemukan dua belah pihak yang melakukan konflik. Media ini digunakan untuk berkomunikasi untuk menyelesaikan konflik. Akan tetapi penyelesaian konflik dengan cara ini masih membutuhkan saran dimana letak terjadinya konflik pada model.

Gomes (Gomes dkk, 2000) melakukan penelitan tentang pengembangan perangkat lunak dengan menggunakan desain perangkat lunak yang sudah ada. Gomes menggunakan ontologi pada WordNet untuk mencari kemiripan objek-

objek diagram kelas antara diagram kelas lama dengan diagram kelas yang akan dibuat.

Dari studi kepustakaan yang telah dilakukan pada beberapa penelitian sebelumnya mengenai pendeteksian konflik pada model, menjadi dasar kontribusi pada penelitian ini untuk membangun sistem pendeteksian konflik dengan menggunakan metode modifikasi graf dan similaritas WordNet. Penggabungan dua metode ini dapat menyelesaikan pendeteksian konflik secara sintaksis dan semantik secara leksikal.

1.2 Perumusan Masalah

Dalam penelitian ini, masalah-masalah yang diselesaikan berdasarkan hasil kajian pustaka yang telah dilakukan dirumuskan sebagai berikut.

1. Bagaimana cara melakukan pendeteksian konflik semantik secara leksikal?
2. Bagaimana cara untuk mengintegrasikan pendekatan modifikasi graf dan similaritas WordNet?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut.

1. Bahasa alami yang dipakai dalam mengembangkan pendekatan ini adalah bahasa Inggris.
2. Studi kasus yang digunakan dalam penelitian ini adalah diagram kelas.
3. Penelitian ini digunakan untuk diagram kelas yang mengalami jenis perubahan berupa penambahan dan penghapusan (*operation-base conflict*).
4. Pengubahan dua orang pengguna terhadap model berawal dari versi yang sama.

1.4 Tujuan dan Manfaat Penelitian

Tujuan penelitian ini adalah sebagai berikut.

1. Mengusulkan metode pendeteksian konflik leksikal pada diagram kelas di lingkungan pengembangan banyak pengguna. Metode tersebut bermanfaat

bagi pemodel yang menggunakan VCS sebagai media penyimpanan modelnya.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini dibahas beberapa teori dasar yang menunjang dalam pembuatan tesis. Di antaranya adalah penelitian-penelitian pendeteksian konflik terdahulu.

2.1 Deteksi Konflik Pada Model

Metode pendeteksian konflik dari penelitian sebelumnya mempunyai kekurangan dan kelebihan masing-masing. Dari hasil studi pustaka yang sudah dilakukan diperoleh perbandingan metode pendeteksi konflik model pada Tabel 2.1

Tabel 2.1 Penelitian Terkait Pendeteksian Konflik Pada Model

	Peneliti		
	Taentzer [2010]	Ehrig [2001]	Rajbhoj [2013]
Model	Diagram Tahap	Diagram Tahap	Metamodel
Metode	Transformasi graf dengan pendekatan DPO (*Double Pushout*)	Transformasi graf dengan pendekatan DPO	Pola graf
Fokus masalah	Sintaksis konflik	Sintaksis konflik	Sintaksis konflik

Ketiga peneliti pada Tabel 2.1 sama-sama menggunakan graf dalam melakukan pendeteksian konflik dan fokus masalah hanya untuk penyelesaian konflik secara sintaksis.

2.2 Kategorisasi Konflik

Konflik dibagi menjadi tiga kategori yaitu konflik tekstual, konflik sintaksis dan konflik semantik (Altmanninger, 2011). Berikut ini adalah penjelasan dari masing-masing kategori konflik tersebut.

A. Tekstual

Konflik tekstual dapat terjadi jika perbandingan dilakukan pada representasi tekstual dari artefak. Unsur-unsur yang dibandingkan dalam pendeteksian konflik secara tekstual adalah baris teks, paragraf, kalimat, kata, atau karakter. Pada Gambar 2.1 adalah potongan XMI yang merupakan representasi tekstual dari diagram kelas. Diagram kelas tersebut mempunyai kelas *Person* dan atribut *birthday*. Pada versi sebelah kiri nama atribut *birthday* diganti dengan *dateOfBirth* dan terjadi penambahan atribut *name*. Pada versi sebelah kanan tipe atribut birthday diganti dari *String* menjadi *Edate* dan terjadi penambahan atribut *lastName*. Penambahan dua atribut *name* dan *lastName* akan menyebabkan konflik karena kedua atribut tersebut berada pada baris yang sama. Pendeteksian konflik dilakukan dengan cara membandingkan baris yang ada pada representasi tekstual. Penambahan atribut *name* dan *lastName* seharusnya tidak menyebabkan konflik karena *name* dan *lastName* adalah penambahan atribut yang berbeda meskipun pada representasi tekstual berada pada baris yang sama dan representasi tekstual dengan menggunakan XMI tidak memperhatikan urutan baris.

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1"
xmlns:xmi=http://schema.omg.org/spec/XMI/2.1>
  <packagedEmelent name="Person">
    <ownedAttribute name="birthday">
    </ownedAttribute>
  </packageElement>
</uml:Package>
```

Versi 0

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1"
xmlns:xmi=http://schema.omg.org/spec/XMI/2.1>
  <packagedEmelent name="Person">
    <ownedAttribute name="dateOfBirth" type="String">
    </ownedAttribute>
    <ownedAttribute name="name">
    </ownedAttribute>
  </packageElement>
</uml:Package>
```

Versi 1

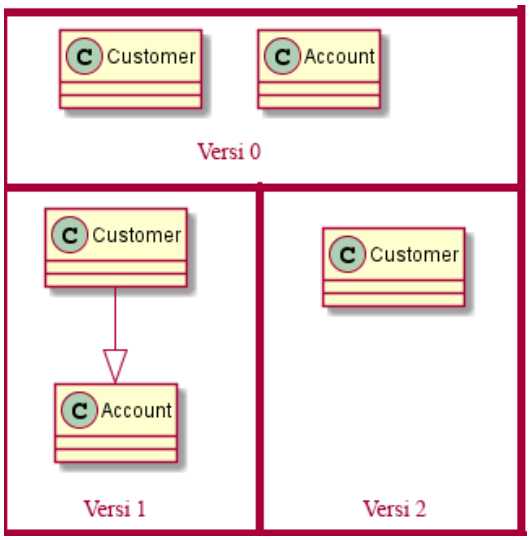
```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1"
xmlns:xmi=http://schema.omg.org/spec/XMI/2.1>
  <packagedEmelent name="Person">
    <ownedAttribute name="birthday" type="EDate">
    </ownedAttribute>
    <ownedAttribute name="lastName">
    </ownedAttribute>
  </packageElement>
</uml:Package>
```

Versi 2

Gambar 2.1 Konflik Tekstual

B. Sintaksis

Konflik sintaksis adalah konflik yang dapat dideteksi dengan cara membandingkan struktur dari artefak. Misalnya dua orang pengguna melakukan modifikasi nama pada elemen yang sama dan modifikasi tersebut dilakukan dengan cara yang berlawanan. Pendeteksian konflik secara sintaksis membutuhkan informasi dari metamodel, konsep bahasa dan relasi. Gambar 2.2 adalah sebuah diagram kelas. Pada versi 1 ditambahkan relasi asosiasi antara kelas *Customer* dan kelas *Account*. Pada versi 2 kelas *Account* dihapus. Setelah dilakukan perbandingan secara struktural berdasarkan sintaks dari artefak model, terdeteksi konflik. Kelas *Account* yang dipakai pada versi 1 dihapus pada versi 2. Kedua versi 1 dan versi 2 tidak bisa digabungkan tanpa adanya interaksi dari dua pengguna yang melakukan modifikasi.



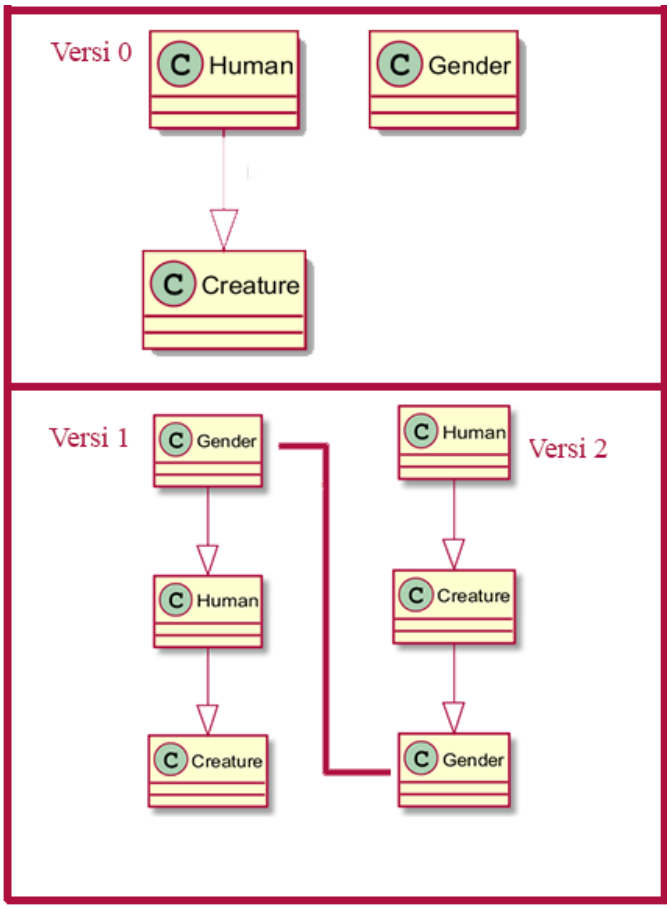
Gambar 2.2 Konflik Sintaksis

C. Semantik

Konflik semantik tidak bisa dideteksi dengan menggunakan pendekatan secara sintaksis ataupun leksikal. Bahkan modifikasi pada artefak yang dilakukan secara bersama-sama akan menghasilkan kebenaran secara sintaksis meskipun modifikasi yang dilakukan saling bertentangan. Ada beberapa macam konflik semantik adalah sebagai berikut :

- Kontradiksi Statik

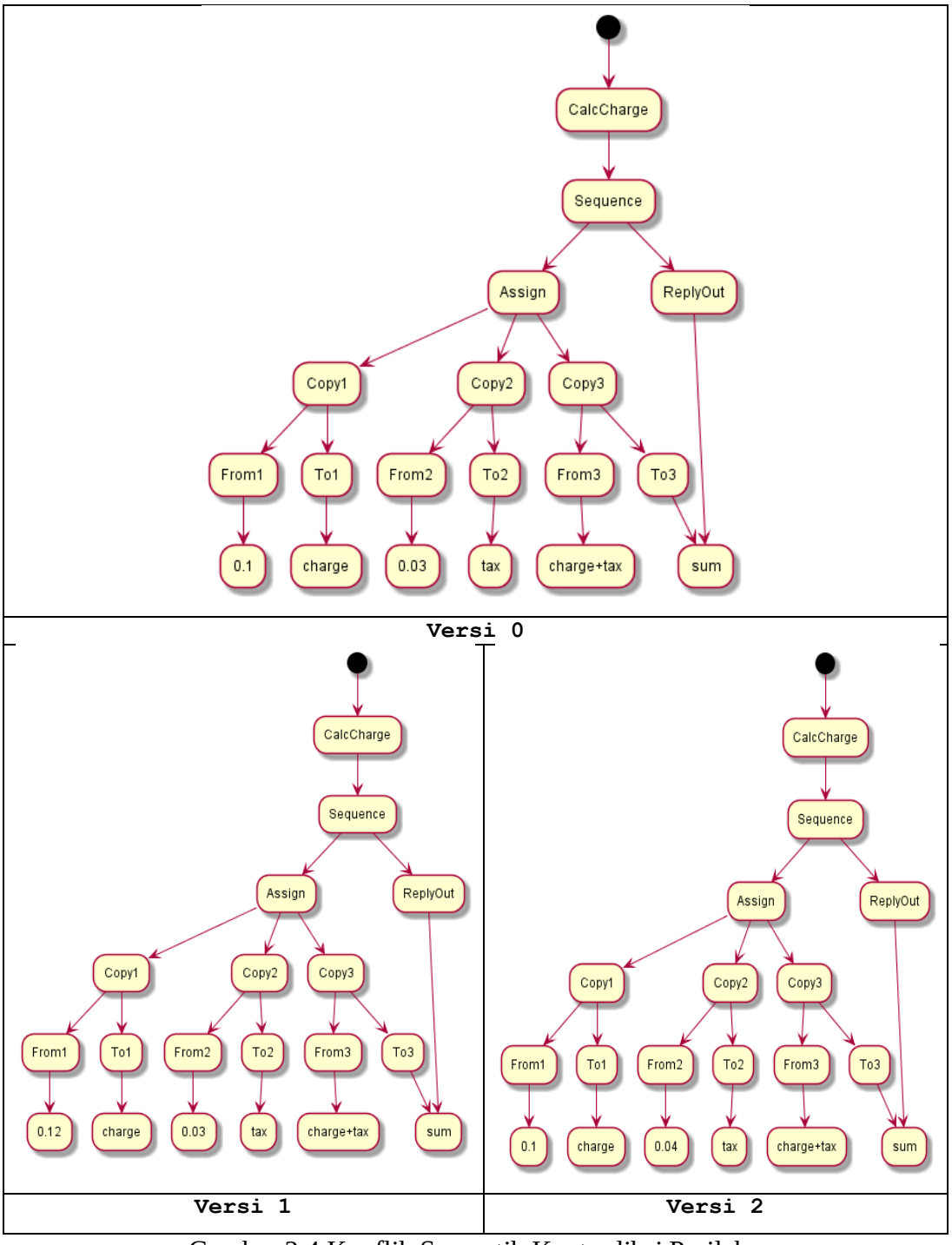
  Konflik semantik kontradiksi statik terjadi karena adanya pelanggaran dari aturan bahasa ketika dilakukan penggabungan pada dua buah model yang terjadi konflik. Aturan pelanggaran ini biasanya terjadi pada model, relasi, konsistensi, dan kondisi tertentu. Pada Gambar 2.3 relasi generalisasi ditambahkan pada kelas *gender*. Hasil penggabungan dari kedua buah model hasil modifikasi akan menghasilkan pewarisan melingkar. Hal ini tidak diperbolehkan dalam diagram kelas karena melanggar aturan secara bahasa pemodelan.



Gambar 2.3 Konflik Semantik Kontradiksi Statik

- Kontradiksi Perilaku

  Salah satu cara pendeteksian konflik semantik adalah dengan memperhatikan perilaku perubahan pada model. Perilaku ini merupakan aktifitas penambahan, penghapusan dan pengubahan pada suatu elemen pada model. Pada Gambar 2.4 adalah model WSBPEL yang menggambarkan perhitungan biaya toko online. Pada versi 1 nilai variabel *charge* diubah dari 0.1 menjadi 0.12. pada versi 2 nilai variabel *tax* diubah dari 0.03 menjadi 004. Pada kedua versi variabel sum diperoleh dengan menjumlahkan nilai *charge* dan *tax*.

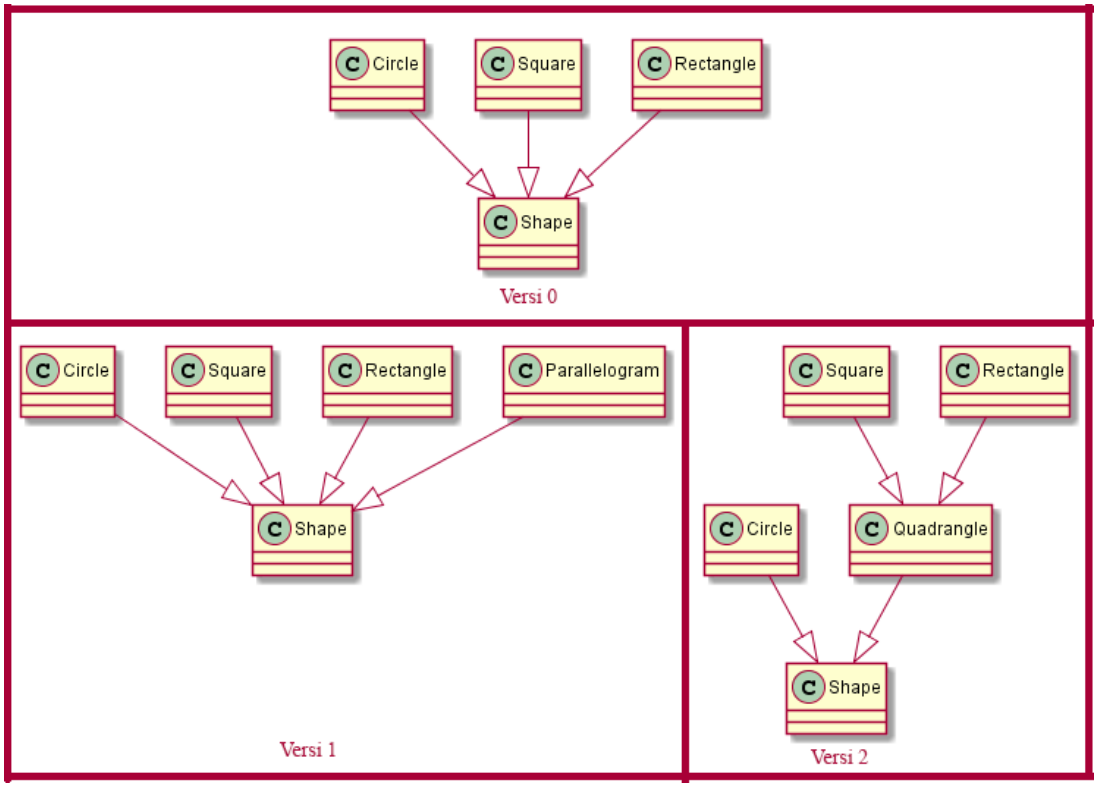


Gambar 2.4 Konflik Semantik Kontradiksi Perilaku

- Leksikal / Kontradiksi Arti

  Konflik semantik leksikal terjadi ketika model mengalami perubahan dengan bahasa yang lain tetapi memiliki arti yang sama. Pada Gambar 2.5 kelas *shape* terdiri dari kelas *circle*, *square* dan *rectangle*. Pada versi 1 kelas *parallelogram* ditambahkan. Pada versi 2 kelas *quadrangle* ditambahkan.

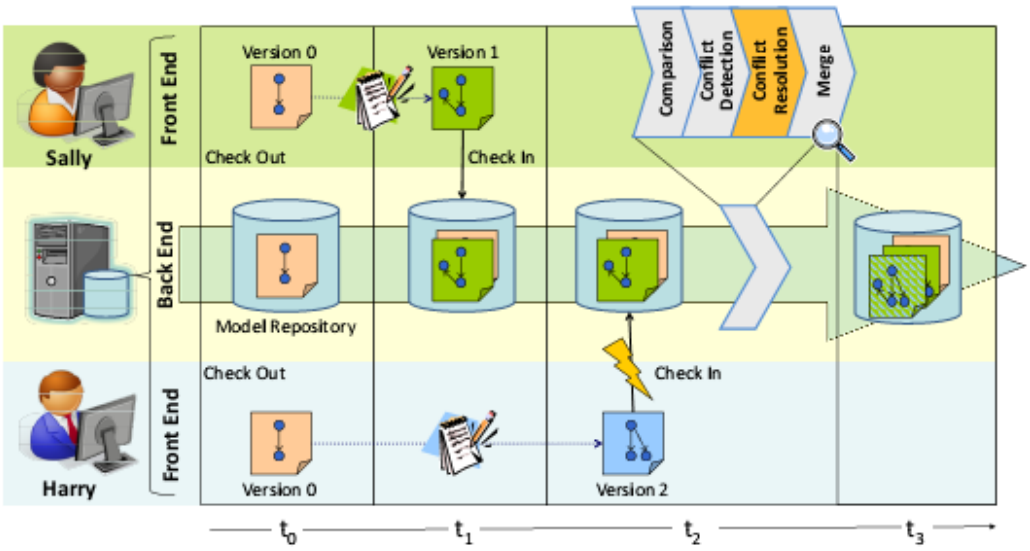
Pada versi penggabungan kelas *parallelogram* dan kelas *quadrangle* akan dianggap sebagai kelas yang berbeda. Padahal kedua kelas tersebut mempunyai arti yang sama.



Gambar 2.5 Konflik Semantik Kontradiksi Arti

2.3 Version Control System (VCS)

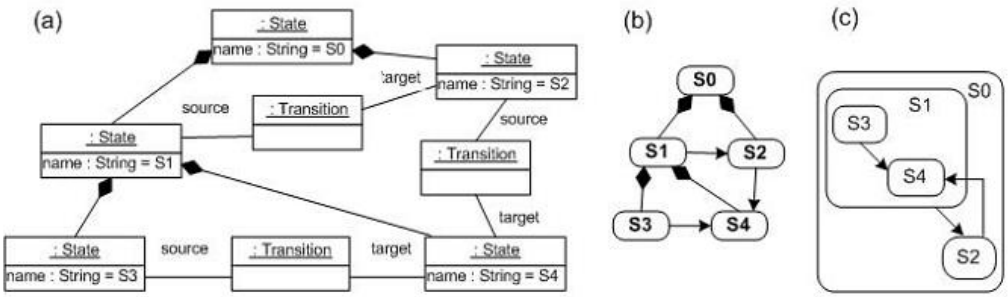
VCS adalah suatu perangkat lunak yang digunakan sebagai pengontrol versi yang dapat mengatur proses pengembangan perangkat lunak yang dilakukan oleh suatu kelompok. Gambar 2.6 merupakan contoh konflik yang terjadi pada VCS dokumen. Dokumen versi yang ke 0 sama-sama diedit oleh sally dan harray. Kemudian sally mengirim dokumen yang sudah diedit (dokumen versi 1) kedalam server. Harry yang sudah terlanjur mengedit dokumen versi 0 akan mendapatkan peringatan dari VCS karena ada versi terbaru yang dibuat oleh sally. Dan akhirnya konflik terjadi antara versi 1 yang dibuat oleh sally dan versi 2 yang dibuat harry. Dalam hal ini harry harus memperbaiki dokumen untuk disesuaikan dengan versi terbaru milik sally.



Gambar 2.6 Konflik pada VCS

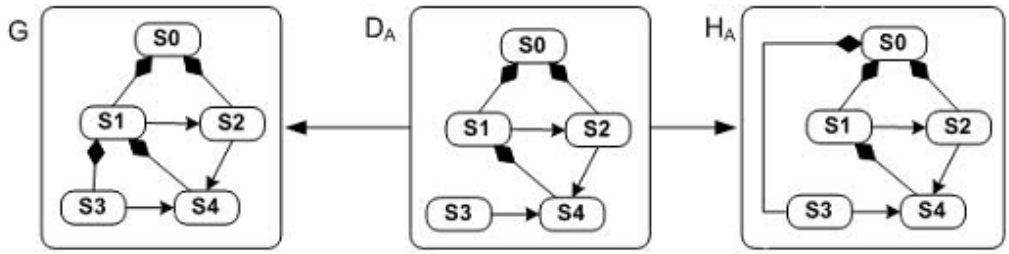
2.4 Modifikasi Graf

Modifikasi graf adalah pendeketan dengan mentransformasikan model dan atributnya kedalam bentuk graf. Misal pada sebuah diagram *statechart*, pada awalnya model ditransformasikan kedalam bentuk *abstract syntax* gambar Gambar 2.7 a. Dalam setiap simpul terdiri dari tipe dan atribut. Sebuah *state* dalam *statechart* diagram digambarkan menjadi sebuah simpul dengan tipe *state*. Setiap simpul mempunyai atribut S0, S1, S2, S3, S4. Sedangkan aliran aktivitas pada diagram *statechart* digambarkan dengan simpul dengan tipe transisi. Kemudian dari abstract syntax ditransformasikan kedalam *compact notation* Gambar 2.7 b. Dan kemudian ditransformasikan kedalam *concrete syntax* (graf) Gambar 2.7 c.



Gambar 2.7 Modifikasi Graf (Taentzer, 2010)

Definisi dari modifikasi graf, misal dua buah graf G dan H, dengan G adalah graf sebelum mengalami perubahan dan H adalah graf setelah mengalami perubahan, dengan $G \Rightarrow H$ adalah perubahan injektif dari $G \overset{g}{\leftarrow} D \overset{h}{\rightarrow} H$. Secara berurutan $G = G_0 \Rightarrow G_1 \Rightarrow ... \Rightarrow G_n = H$ adalah modifkasi graf secara langsung dan dinotasikan dengan $G \overset{*}{\Rightarrow} H$. Dimana graf D adalah graf penengah yang berisi semua penghapusan graf yang sudah dilakukan dan tidak ada penambahan.



Gambar 2.8 Transformasi Graf (Taentzer, 2010)

Pada Gambar 2.8 seorang pengguna mengubah sebuah statechart diagram dari G ke H_A dengan memasukkan S3 kedalam S0 dan melakukan penghapusan S3 dari S1. Graf D_A adalah graf yang menunjukkan proses penghapusan S3 dari S1. Sedangkan graf H_A adalah graf yang menunjukkan proses penambahan S3 kedalam S0.

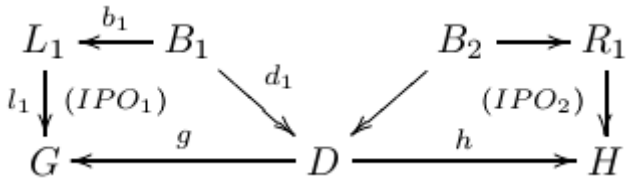
2.5 Transformasi Graf

Graf yang mengalami perubahan atau transformasi dapat dicari dengan menggunakan pendekatan *double pushout* (DPO). Pendekatan DPO ini mengggunakan operasi penghapusan dan penambahan pada graf. Pencarian bagian mana yang mengalami perubahan dilakukan dengan cara mencari aturan minimal. Aturan minimal adalah bagian terkecil (atom) dari sebuah graf yang mengalami perubahan.

Langkah-langkah mencari aturan minimal pada sebuah graf adalah sebagai berikut:

1. Inisialisasi *pushout*

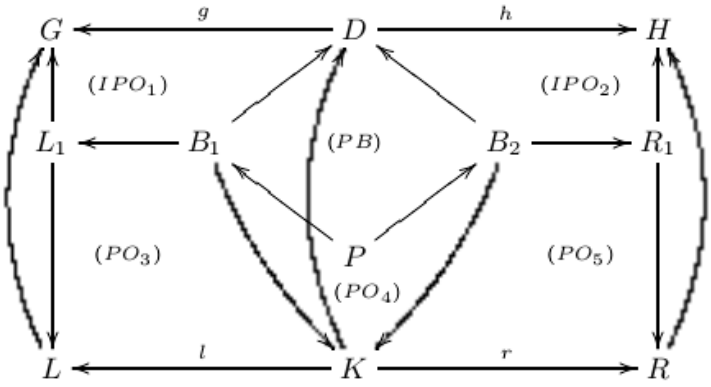
   Misal $g: D \rightarrow G$ adalah perubahan graf. Inisial *pushout* berisi perubahan graf $l_1: L_1 \rightarrow G$, $b_1: B_1 \rightarrow L_1$ dan $d_1: B_1 \rightarrow D$ seperti pada Gambar 2.9.



Gambar 2.9 Inisial pushout

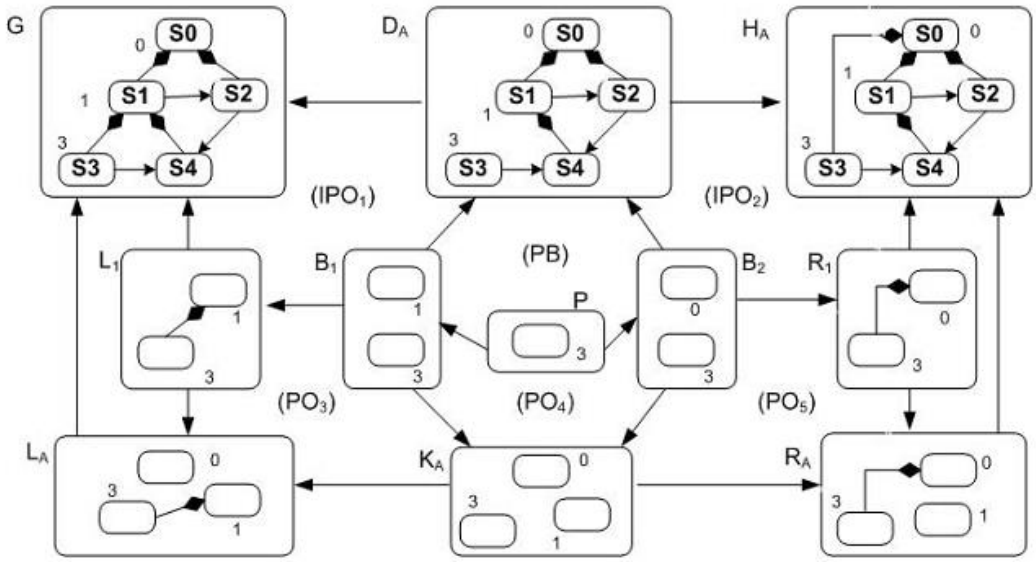
2. Mendefinisikan $B_1 \leftarrow P \rightarrow B_2$ sebagai *pullback* dari $B_1 \leftarrow D \rightarrow B_2$ dan $B_1 \leftarrow K \rightarrow B_2$ sebagai *pushout* (PO_4) dari $B_1 \leftarrow P \rightarrow B_2$ dengan induksi perubahan $K \rightarrow D$
3. Bangun $B_1 \leftarrow L \rightarrow K$ sebagai *pushout* (PO_3) dari $L_1 \leftarrow B_1 \rightarrow K$ dengan induksi perubahan $L \rightarrow G$. Bangun $R_1 \leftarrow R \rightarrow K$ sebagai *pushout* (PO_5) dari $R_1 \leftarrow B_2 \rightarrow K$ dengan induksi perubahan $R \rightarrow H$
4. Karena IPO_1 dan PO_3 adalah *pushout* maka (IPO_1) + (PO_3) adalah *pushout*. Dan karena IPO_2 dan PO_5 adalah *pushout* maka (IPO_2) + (PO_5) adalah *pushout*.

Hasil akhir kontruksi aturan minimal bisa dilihat pada Gambar 2.10



Gambar 2.10 Kontruksi Aturan Minimal

Kontruksi aturan minimal pada studi kasus Gambar 2.8 bisa dilihat pada Gambar 2.11



Gambar 2.11 Aturan Minimal $\boldsymbol{P_A = (L_A \leftarrow K_A \rightarrow R_A)}$

2.6 Karakteristik konflik

Ehrig (Ehrig dkk, 2011), dua buah modifikasi graf m_1 dan m_2 dikatakan konflik jika memenuhi salah satu persyaratan sebagai berikut :

1. Kedua modifikasi graf tersebut menghapus elemen yang sama.
2. m_1 menghapus simpul yang akan menjadi sumber atau tujuan dari simpul baru yang ditambahkan m_2.
3. m_2 menghapus simpul yang akan menjadi sumber atau tujuan dari simpul baru yang ditambahkan m_3.

Jika $m_i = \left(G \overset{D_i}{\Rightarrow} H_i\right) (i = 1,2)$, maka (m_1, m_2) dikatakan konflik jika

1. (m_1, m_2) dalam konflik *delete-delete,* atau
2. (m_1, m_2) dalam konflik *delete-insert*, atau
3. (m_2, m_1) dalam konflik *delete-insert*.

2.7 Similaritas WordNet

WordNet adalah suatu sistem referensi leksikal bahasa inggris yang bersifat online. WordNet dikembangkan oleh Cognitive Science Laboratory di Universitas Princeton yang dikepalai oleh George Miller. Arti dari suatu kata pada WordNet direpresentasikan dengan *synonym sets* (*synsets*). Synsets adalah

daftar *term* atau *collocation* yang artinya sama dan dalam konteks tertentu penggunaannya dapat saling dipertukarkan. Dalam *synset* juga dicatat pointer-pointer ke *synset* lain yang digunakan untuk mendeskripsikan relasi antar *synset*. WordNet dibagi dalam empat taksonomi berdasarkan tipe kata yaitu kata benda, kata kerja, kata keterangan, dan kata sifat (Miller, 1993).

Macam-macam hubungan antar kata pada WordNet adalah sebagai berikut

- *Polysemy*

  Polysemy adalah kata yang mempunyai banyak arti. Misal kata plain mempunyai arti sederhana dalam kalimat "*English is extremely plain subject*" dan mempunya arti tidak ada suatu yang ditambahkan pada kalimat "*This blouse is too plain*".
- *Synonimy*

  Synonomy adalah dua buah kata yang berbeda dalam tulisan dan pengucapan tetapi mempunyai arti yang sama. Misal kata *small* dengan *little* dan kata *big* dan *large*.
- *Antonymy*

  Antonymy adalah dua buah kata yang mempunya pertentangan arti. Misal kata *big* dengan *small* dan kata *good* dengan *bad*.
- *Hyponymy*

  Hyponymy adalah dua buah kata yang mempunyai hubungan umum-khusus. Misal kata *red* dengan kata *color, red* adalah *hyponymy* dari *color*.
- *Meronymy*

  Meronymy adalah dua buah kata yang mempunyai relasi hubungan bagian. Misal kata *finger* dengan *hand, finger* adalah bagian dari *hand*.
- *Idiom*

  Idiom adalah kumpulan kata yang mempunyai arti bukan sebenarnya.

2.7.1. Similaritas antar kelas

Pada penelitian ini model yang digunakan sebagai studi kasus adalah diagram kelas. Pencarian persamaan diagram kelas versi satu dengan yang lain menggunakan matrik berikut.

$$S(C_1, C_2) = \begin{bmatrix} \omega_1 . S(S_1, S_2) & + \\ \omega_2 . S(I_{e1}, I_{e2}) & + \\ \omega_3 . S(I_{a1}, I_{a2}) & \end{bmatrix} \quad (1)$$

Dimana $S(C_1, C_2)$ adalah sebuah kelas, kelas C_1 dan kelas C_2, $S(S_1, S_2)$ adalah tingkat kemiripan kategori dari atribut diagram kelas atau nama kelas, $S(I_{e1}, I_{e2})$ adalah tingkat kemiripan relasi antara kelas C_1 dan kelas C_2, $S(I_{a1}, I_{a2})$ adalah tingkat kemiripan atribut dan metode antara kelas C_1 dan kelas C_2. $\omega_1, \omega_2, \omega_3$ adalah konstanta tingkat kemiripan dimana ω_1 adalah 0,6, ω_2 adalah 0,1 dan ω_3 adalah 0,3. Nilai konstanta dari ω diperoleh dari hasil penelitian sebelumnya (Paulo dkk, 2000).

2.7.2. Menghitung Similaritas WordNet

Wu & Palmer (Wup) menghitung similaritas dua synset dalam WordNet berdasarkan kedalaman dua synset tersebut dalam taksonomi WordNet. Pencarian similaritas antara dua synset menggunakan persamaan berikut.

$$wup_{similarity}(S_1, S_2) = \frac{2\ x\ depth(lcs(S_1, S_2))}{depth(S_1) + depth(S_2)} \quad (2)$$

dengan S_1, S_2 adalah *synset* yang dibandingkan, *depth* adalah kedalaman *synset* dalam hirarki, dan lcs (*least common subsumer*) adalah konsep umum.

$$SSC(C_1, C_2) = \big(PL(C_1, C_2) + WuP(C_1, C_2)\big)/2 \quad (3)$$

Kemiripan semantik antara dua buah kelas dihitung berdasarkan kemiripan nama kelas, atribut, operasi, dan kemiripan relasi dengan tetangga. Matrik kemiripan semantik antara dua buah kelas C_1 dan C_2 dihitung dengan menggunakan persamaan berikut

$$NS(C_1, C_2) = SSC\big(Name(C_1), Name(C_2)\big) \quad (4)$$

Matrik kemiripan antara dua set atribut, A_1 dan A_2, dari dua buah kelas, C_1 dan C_2, dihitung dengan menggunakan persamaan berikut

$$ASim(C_1, C_2) = Max\left[\forall \sum_{k=1}^{|A_1|} aSim(a_k, a_l)\right] / |A_2| \quad (5)$$

dimana $a_k \in A_1$ dan $a_l \in A_2$, $|A_1| \leq |A_2|$.

Matrik kemiripan antara dua set operasi, O_1 dan O_2, dari dua buah kelas, C_1 dan C_2, dihitung dengan menggunakan persamaan berikut

$$OSim(C_1, C_2) = Max\left[\forall \sum_{k=1}^{|O_1|} oSim(o_k, o_l)\right] / |O_2| \quad (6)$$

dimana $o_k \in O_1$ dan $o_l \in O_2$, $|O_1| \leq |O_2|$.

Matrik kemiripan internal (Semantik) adalah kemiripan antar atribut diagram kelas seperti nama atribut dan nama operasi. Matrik kemiripan dihitung dengan menggunakan persamaan sebagai berikut :

$$IS(C_1, C_2) = W_a \times ASim(C_1, C_2) + W_m \times OSim(C_1, C_2) \quad (7)$$

dimana W_a merupakan bobot kemiripan atribut dan W_m adalah bobot kemiripan operasi.

Misal dua buah diagram kelas M1 dan M2 dengan kelas-kelas penyusun C_1, C_2, C_3, C_{dst} akan dicari tingkat kemiripannya. Tabel 2.2 sampai dengan Tabel 2.5 merupakan ilustrasi memasangkan kelas dengan tingkat kemiripan tertinggi hingga terendah.

Tabel 2.2 Matrik Kemiripan Diagram Kelas M1 dan M2

		M2			
		C_1	C_2	C_3	C_4
M1	C_1	0.48	0.55	0.35	0.6
	C_2	0.57	0.5	0.55	0.89
	C_3	0.95	0.6	0.61	0.54

Tabel 2.3 Matrik Kemiripan Dengan Nilai Tertinggi

		M2			
		C_1	C_2	C_3	C_4
M1	C_1	0.48	0.55	0.35	0.6
	C_2	0.57	0.5	0.55	0.89
	C_3	**0.95**	0.6	0.61	0.54

Pada Tabel 2.3 kelas C3 dan C1 merupakan kelas yang mempunyai persamaan kata dengan tingkat kemiripan tertinggi.

Tabel 2.4 Matrik Kemiripan Tertinggi Iterasi Kedua

		M2			
		~~C_1~~	C_2	C_3	C_4
M1	C_1	0.48	0.55	0.35	0.6
	C_2	0.57	0.5	0.55	**0.89**
	~~C_3~~	0.95	0.6	0.61	0.54

C2 dan C4 merupakan kelas yang mempunyai persamaan kata dengan nilai kemiripan tertinggi kedua.

Tabel 2.5 Matrik Kemiripan Tertinggi Iterasi Terakhir

		M2			
		~~C_1~~	C_2	C_3	~~C_4~~
M1	C_1	0.48	**0.55**	0.35	0.6
	~~C_2~~	0.57	0.5	0.55	0.89
	~~C_3~~	0.95	0.6	0.61	0.54

2.8 Konvensi Kode Java

Konvensi kode pada java dibuat agar kode program lebih mudah dimengerti dan lebih mudah dibaca. Konvensi kode pada java telah ditetapkan oleh *Sun Microsystems, Inc.* seperti pada Tabel 2.6. Konvensi Penamaan pada *java*. (Sun, 1997)

Tabel 2.6 Konvensi Penamaan Pada Java

Tipe *Indentifier*	Aturan Penamaan	Contoh
Classes	Nama kelas diberi nama kata benda, apabila lebih dari satu kata dengan huruf pertama dari setiap kata menggunakan huruf kapital. gunakan nama kelas dengan nama yang sederhana dan deskriptif. Gunakan seluruh kata-hindari akronim dan singkatan (kecuali singkatan yang jauh lebih banyak digunakan daripada bentuk panjang, seperti URL atau HTML).	class Raster; class ImageSprite; class UserStory;

Interfaces	Nama *interface* sama dengan nama kelas dengan huruf capital pada huruf pertama setiap kata	interface RasterDelegate; interface Storing;

Methods	Metode harus diberi nama dengan kata kerja, apabila lebih dari satu kata gunakan huruf kecil pada kata pertama, gunakan huruf capital pada setiap huruh pertama dari setiap kata berikutnya	run();
		runFast();
		getBackground();

Variables	Nama variabel, semua *instance*, kelas, dan konstanta kelas apabila lebih dari satu kata gunakan huruf kecil pada kata pertama, gunakan huruf capital pada setiap huruh pertama dari setiap kata berikutnya, nama variabel tidak harus dimulai dengan garis bawah _ atau tanda dolar $ karakter, meskipun keduanya diizinkan. Nama variabel harus pendek namun bermakna. Pemilihan nama variabel harus mnemonic- yaitu, dirancang agar pembaca mengerti maksud penggunaannya. Nama variabel satu karakter harus dihindari kecuali untuk sementara ("sekali pakai") variabel. Nama-nama umum variabel sementara contoh *i, j, k, m,* dan *n* untuk bilangan bulat; *c, d,* dan *e* untuk karakter.	int i; char c; float myWidth;

Constants	Nama-nama konstanta kelas variabel yang dideklarasikan dan konstanta ANSI harus semua huruf kapital dengan kata-kata yang dipisahkan oleh garis bawah ("_"). (Konstanta ANSI harus dihindari, untuk kemudahan debugging.)	static final int MIN_WIDTH = 4;
		static final int MAX_WIDTH = 999;
		static final int GET_THE_CPU = 1;

2.9 Koefisien Cohen's Kappa

Koefisien Cohen's Kappa adalah koefisien untuk mengevaluasi kesepakatan antara beberapa penilai atau dua metode penilaian. Koefisien Cohen's Kappa diusulkan oleh Jacob Cohen pada tahun 1960. Koefisien ini dapat dirumuskan menggunakan persamaan berikut (Sim & Wright, 2005) :

$$K = \frac{P_o - P_c}{1 - P_c} \quad (8)$$

dimana, P_o adalah kesepakatan yang terobservasi dan P_c adalah kesepakatan yang diharapkan secara kebetulan. Data yang didapat dari hasil pengamatan dari dua pengamat digambarkan dalam bentuk tabel relevansi pada Tabel 2.7.

Tabel 2.7 Tabel Relevansi

Pengamat		**II**		**Total**
		Relevan	**Tidak Relevan**	
I	**Relevan**	a	b	g_1
	Tidak Relevan	c	d	g_2
Total		f_1	f_2	n

berdasarkan Tabel 2.7, P_o didapatkan dengan menjumlahkan nilai pada a dan d, dan dibagi dengan n. P_o dirumuskan sebagai berikut (Sim & Wright, 2005):

$$P_o = \frac{a + d}{n}$$

dimana a dan d masing-masing adalah kesamaan relevan dan tidak relevan, dan n adalah total data. Sedangkan P_c didapatkan dengan menggunakan rumus berikut ini (Sim & Wright, 2005):

$$P_c = \frac{\left(\frac{f_1 x g_1}{n}\right) + \left(\frac{f_2 x g_2}{n}\right)}{n}$$

sehingga dengan mendapatkan nilai P_o dan P_c, maka nilai K (Cohen's Kappa) dapat dihitung. Nilai kappa dapat menentukan tingkat kesepakatan antar pakar dengan sistem. Nilai koefisien Cohen's Kappa dapat dinterpretasikan pada Tabel 2.8.

Tabel 2.8 Interpretasi Nilai Kappa (Altman, 1991)

Indeks Kappa	**Proporsi Kesepakatan**
< 0.20	Rendah (*Poor*)
0.21 – 0.40	Sedikit (*Fair*)
0.41 – 0.60	Cukup (*Moderate*)
0.61 – 0.80	Kuat (*Good*)
0.81 – 1.00	Sangat Kuat (*Very Good*)

BAB 3
METODE PENELITIAN

Pada penelitian ini, terdapat beberapa tahapan penyelesaian yang akan dilakukan, yang masing-masing tahapan menggunakan suatu metode tertentu. Adapun tahapan dan metode yang digunakan adalah sebagai berikut.

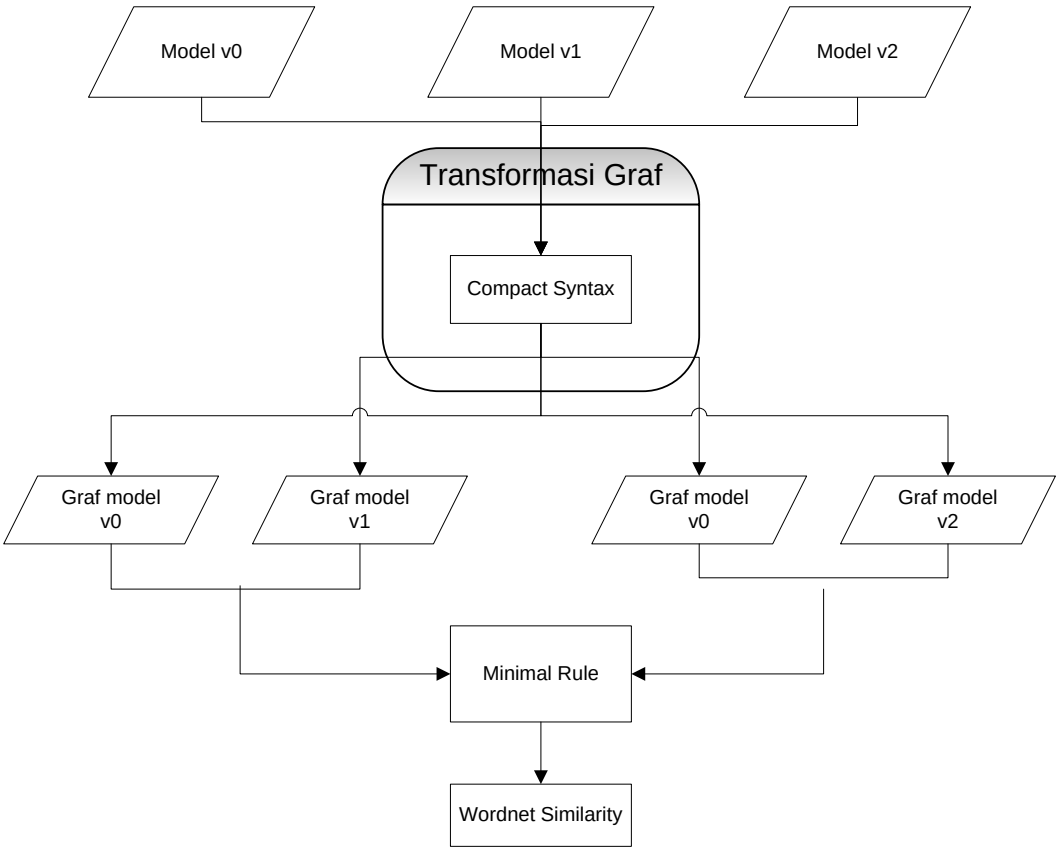
1. Studi literatur
2. Perancangan Sistem
3. Pengembangan Perangkat Lunak
4. Pengujian dan Analisis
5. Penulisan Buku Tesis

3.1 Studi Literatur

Pada bagian ini dipelajari jurnal-jurnal dan buku-buku yang berhubungan dengan teori-teori pengembangan pada lingkungan banyak pengguna pada model, metode pendeteksian konflik dan similaritas WordNet.

3.2 Perancangan Sistem

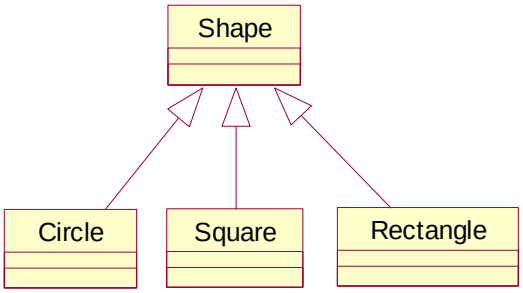
Sistem pendeteksian konflik pada penelitian ini menggunakan dua buah pendekatan yaitu pendekatan modifikasi graf dan similaritas WordNet. Sedangkan model yang digunakan sebagai studi kasus adalah diagram kelas.
Rancangan sistem pendeteksi konflik pada model adalah sebagai berikut.



Gambar 3.1 Metode pendeteksi konflik

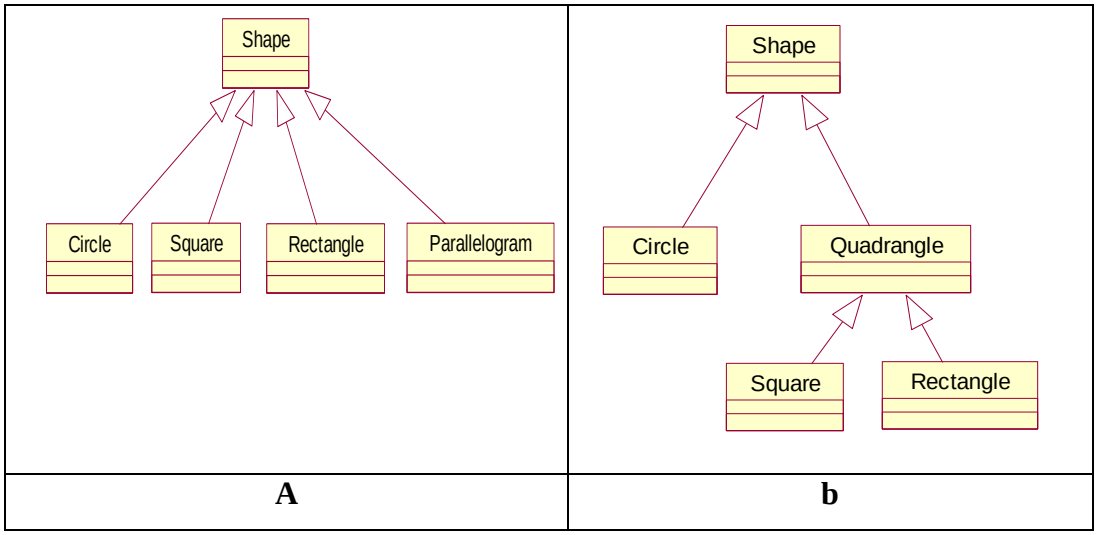
Pada Gambar 3.1 model diagram kelas ditransformasi menjadi bentuk graf. Kemudian graf dari masing-masing versi diagram kelas dibandingkan dengan versi yang ke 0. Proses ini menghasilkan graf dengan aturan minimal. Nilai similaritas WordNet dihitung berdasarkan kemiripan graf dengan aturan minimal versi 1 dan versi 2.

Gambar 3.2 merupakan diagram kelas yang akan digunakan sebagai percobaan studi kasus dalam metode pendeteksian konflik pada penelitian ini.



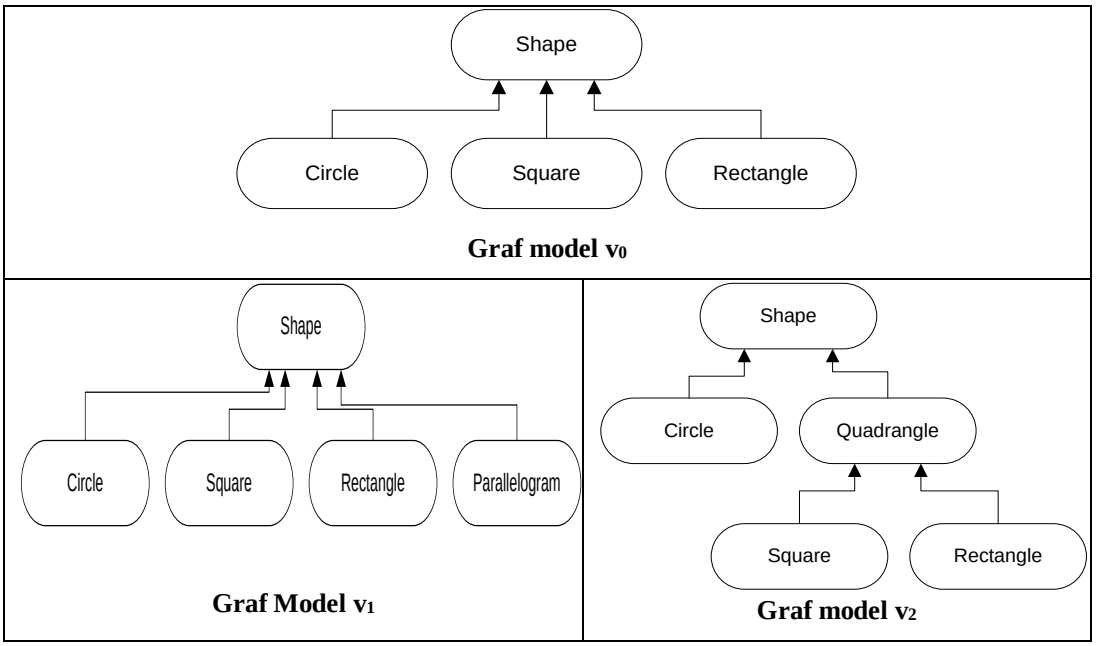
Gambar 3.2 Diagram Kelas versi 0 (v_0)

Dua orang pengguna mengubah diagram kelas v_0 pada gambar 3.1.2.2 sehingga dihasilkan diagram kelas v_1 pada Gambar 3.3 a yang diubah oleh pengguna pertama dan diagram kelas v_2 pada Gambar 3.3 b yang diubah oleh pengguna kedua.



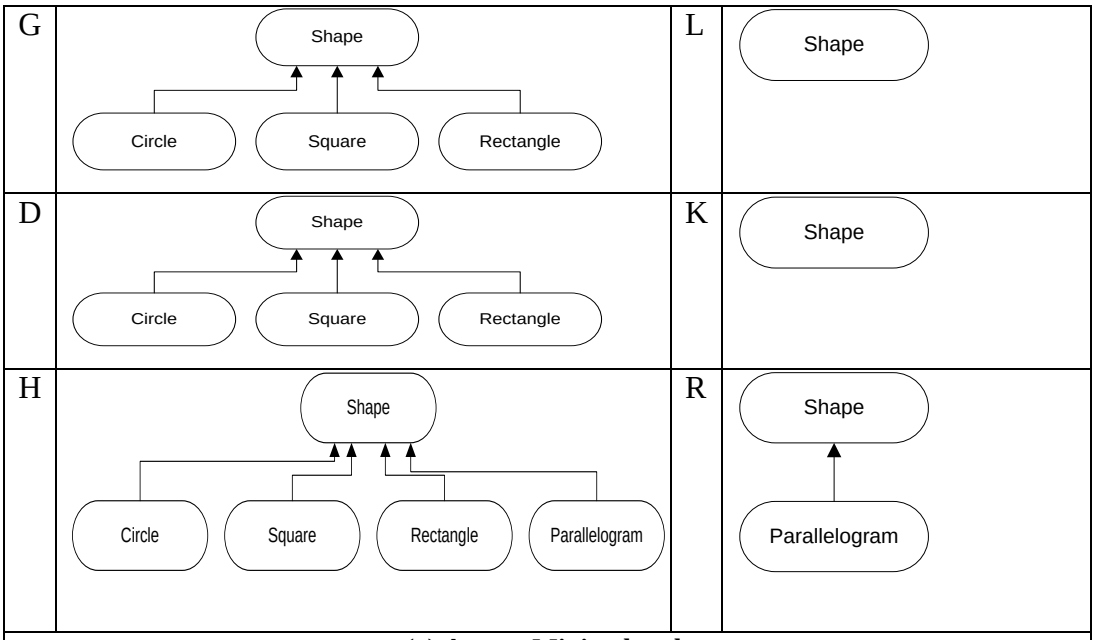
Gambar 3.3 Diagram Kelas hasil modifikasi

Pendeteksian konflik dimulai dengan membandingkan dua buah model diagram kelas v_0 dengan v_1 dan v_0 dengan v_2. Kemudian dua buah versi diagram kelas ini ditransformasikan kedalam bentuk graf melalui transformasi graf kedalam bentuk *abstract syntax*. Atribut-atribut didalam diagram kelas dipetakan kedalam bentuk *abstract syntax*, nama kelas, relasi, atribut dan operasi. Dari *abstract syntax* ini kemudian ditransformasikan kedalam bentuk *compact syntax* sehingga dihasilkan diagram kelas dalam bentuk graf, graf model v_0, v_1 dan graf model v_2 seperti pada Gambar 3.4.

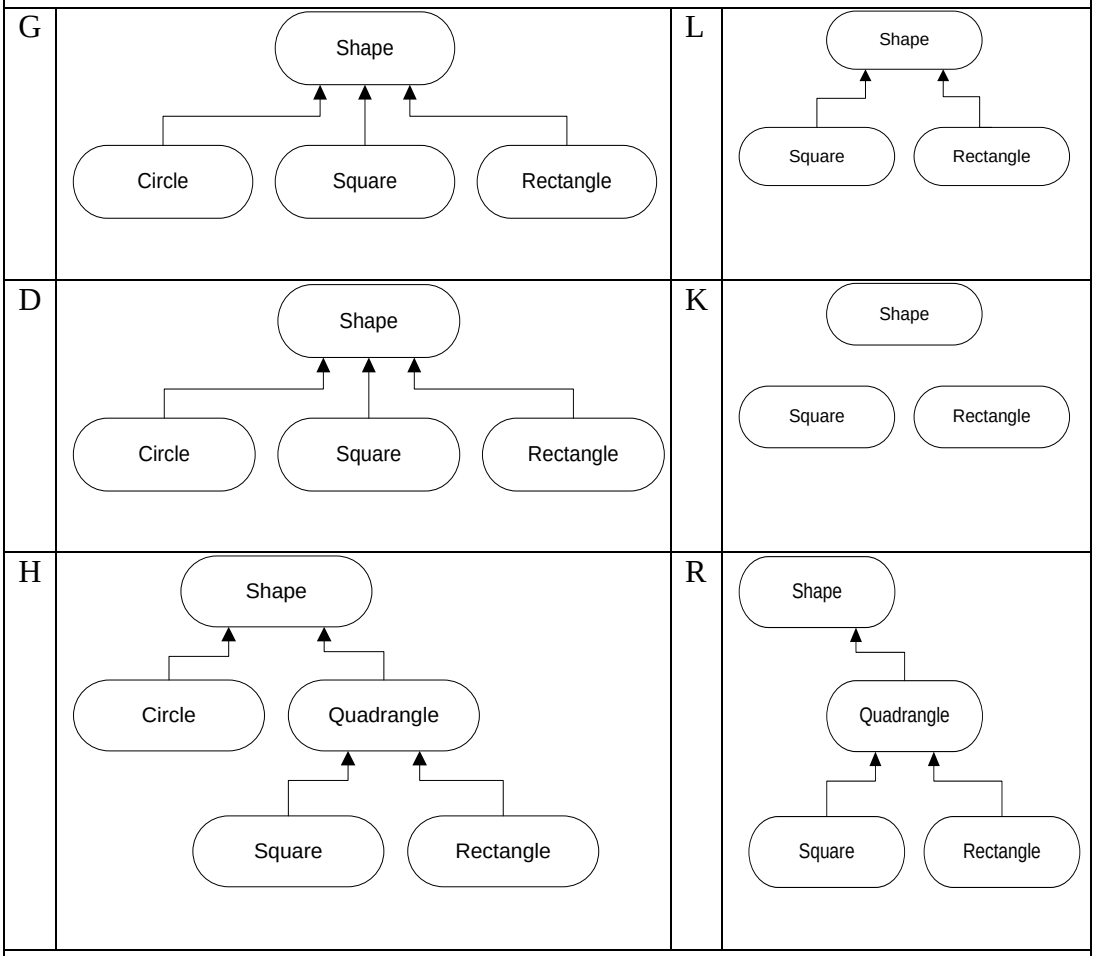


Gambar 3.4 Hasil transformasi kedalam bentuk graf

Graf model v_0 dengan v_1 dan v_0 dengan v_2 dicari aturan minimalnya dengan menggunakan DPO sehingga ditemukan simpul yang mengalami perubahan. Gambar 3.5 a merupakan hasil pencarian aturan minimal dari graf model v_0 dengan v_1 dan Gambar 3.5 b merupakan hasil pencarian aturan minimal dari graf model v_0 dengan v_2. Dengan G adalah graf sebelum mengalami perubahan, H adalah graf setelah mengalami perubahan dan D adalah graf penengah yang berisi semua penghapusan dan tidak ada penambahan. L, K, dan R masing-masing adalah aturan minimal dari graf G, D, dan H.



(a) Aturan Minimal v_0 dan v_1

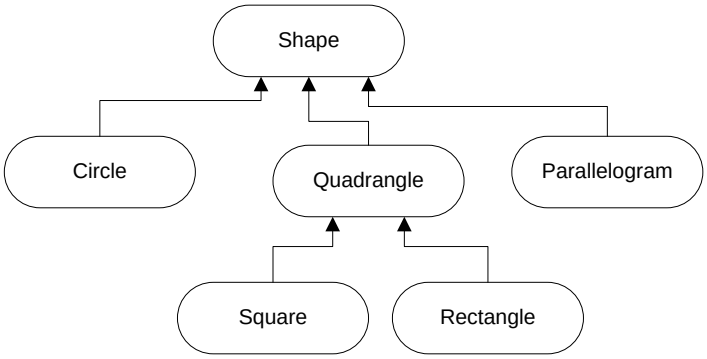


(b) Aturan Minimal v_0 dan v_2

Gambar 3.5 Graf dengan aturan minimal

Kemudian dari hasil pencarian aturan minimal dapat diketahui simpul yang mengalami konflik secara sintaksis. Pencarian simpul yang mengalami konflik secara sintaksis ini dilakukan dengan cara membandingkan simpul yang mengalami penghapusan pada v_1 apakah ada pengubahan pada v_2 dan sebaliknya simpul yang mengalami penghapusan pada v_2 apakah ada pengubahan pada v_1.

Pada contoh studi kasus diatas tidak ditemukan konflik sintaksis karena tidak ada simpul yang mengalami penghapusan tetapi mengalami perubahan pada versi yang lain. Sehingga kedua versi pengubahan tersebut bisa digabungkan seperti pada Gambar 3.6.

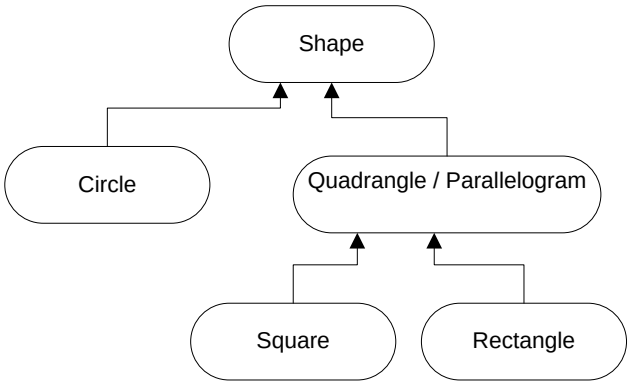


Gambar 3.6 Hasil penggabungan v_1 dan v_2

Dari hasil penggabungan v_1 dan v_2 diatas, kelas *Quadrangle* dan kelas *Parallelogram* adalah dua buah kelas yang mempunyai persamaan makna secara leksikal sehingga mengakibatkan terjadinya konflik semantik leksikal.

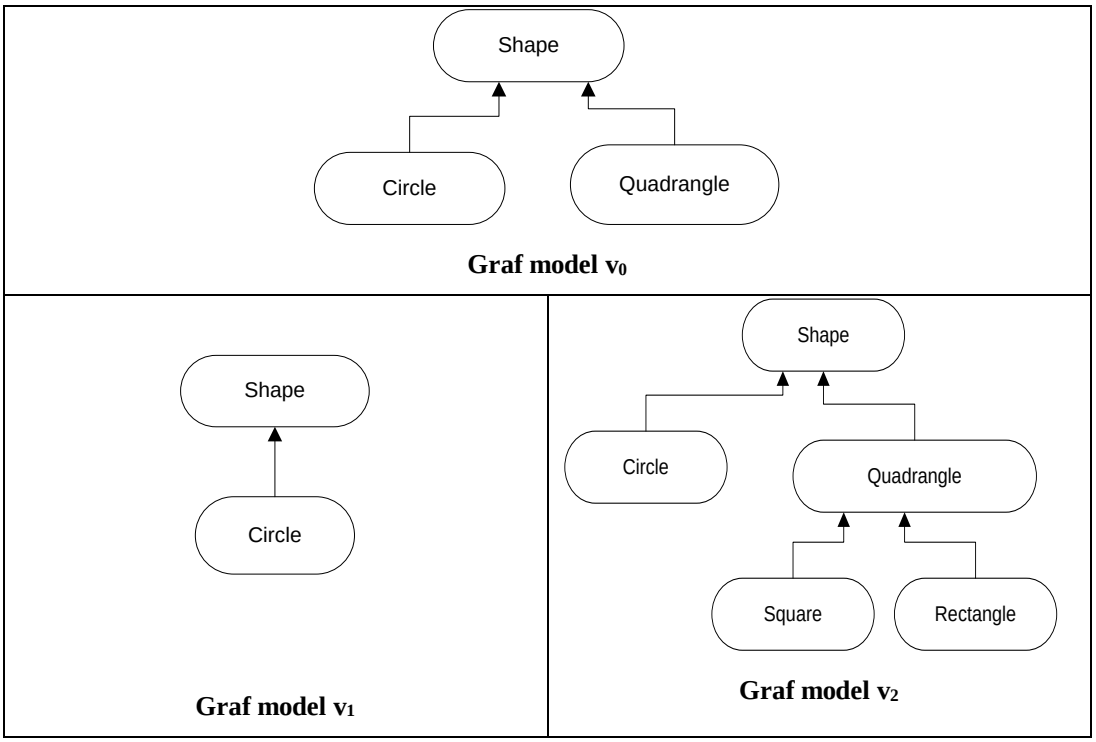
Kontribusi pada penelitian ini adalah menambahkan pendeteksian konflik secara semantik leksikal yang tidak dapat dideteksi dengan menggunakan metode modifikasi graf. Pendeteksian secara semantik leksikal ini dilakukan dengan cara menggunakan similaritas leksikal pada WordNet. Dari hasil pencarian aturan minimal ditemukan simpul baru atau penambahan kelas baru. Simpul baru yang berhasil dideteksi dibandingkan kemiripannya dengan menggunakan similaritas WordNet. Jika terdapat simpul yang memiliki kemiripan tinggi maka simpul tersebut dikatakan sebagai konflik semantik *insert-insert*. Dalam pendeteksian konflik dengan modifikasi graf tidak ada konflik *insert-insert*, karena penambahan simpul baru dengan atribut yang sama dianggap identik secara penulisan tanpa memperhatikan makna.

Dari contoh studi kasus diatas, ditemukan simpul baru *Quadrangle* dan *Parallelogram*. Dengan menggunakan persamaan 2 diperoleh nilai kemiripan antara *Quadrangle* dan *Parallelogram* menggunakan pendeketan Wup adalah sebesar 0,94. *Quadrangle* dan *Parallelogram* memilika nilai kemiripan yang tinggi sehingga bisa dikatakan bahwa *Quadrangle* dan *Parallelogram* adalah simpul dengan makna yang sama. Hasil akhir penggabungan antara v_1 dan v_2 adalah seperti pada Gambar 3.7



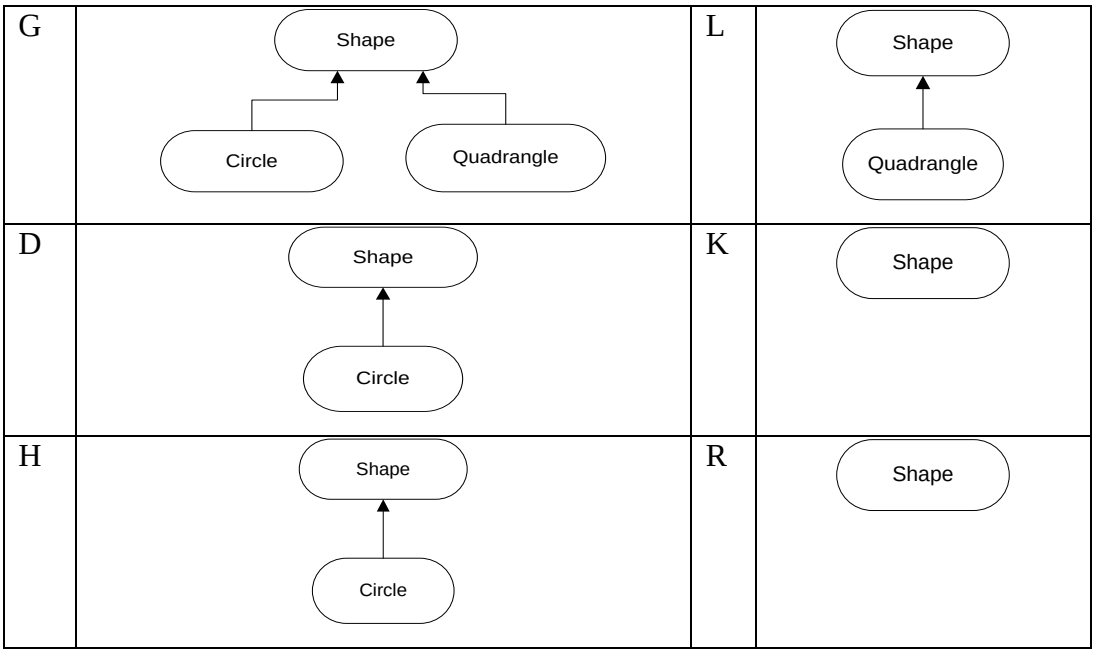
Gambar 3.7 Hasil akhir penggabungan v_1 dan v_2

Pada contoh studi kasus diatas adalah contoh studi kasus untuk konflik secara semantik tetapi tidak konflik secara sintaksis. Pada contoh studi kasus berikut akan dijelaskan konflik yang terjadi secara sintaksis. Pada Gambar 3.8 v_0 adalah graf awal. Kemudian diubah oleh dua orang pengguna menghasilkan v_1 dan v_2.

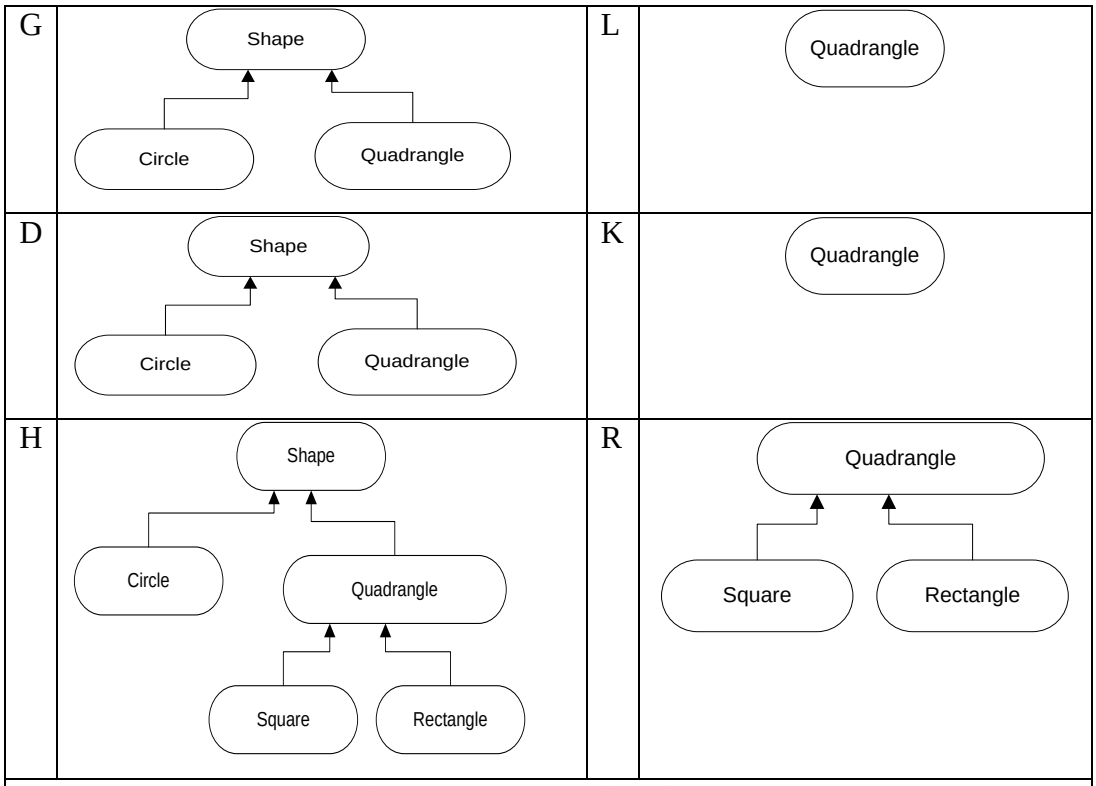


Gambar 3.8 Model Graf

Dari ketiga versi diagram kelas tersebut kemudian dicari aturan minimalnya dengan menggunakan graf modifikasi. Hasil pencarian aturan minimal v_0 dan v_1 dapat dilihat pada Gambar 3.9 sedangkan aturan minimal v_0 dan v_2 dapat dilihat pada Gambar 3.10.



Gambar 3.9 Graf v_0 dan v_1 dengan aturan minimal



(b) Aturan Minimal v_0 dan v_2

Gambar 3.10 Graf v_0 dan v_2 dengan aturan minimal

Dari hasil pencarian aturan minimal diatas dapat diketahui bahwa simpul *Quadrangle* yang masih digunakan pada v_2 dihapus pada v_1. Hal ini menyebabkan terdeteksinya konflik secara sintaksis.

3.3 Pengembangan Perangkat Lunak

Perangkat lunak untuk pendeteksian konflik pada model ini dikembangkan dengan menggunakan bahasa pemrograman java. Pembuatan model graf dengan menggunakan pustaka bantu Jgraph, sedangkan untuk visualisasi diagram kelas dengan menggunakan pustaka Plantuml.

3.4 Pengujian dan Analisis

Pengujian terhadap pendeteksian konflik pada diagram kelas dilakukan dengan menggunakan pengujian kappa yang dilakukan dengan penyebaran kuisioner kepada praktisi dibidang teknologi informasi khususnya dibidang pemodelan diagram kelas.

A. Lingkungan Pengujian

Lingkungan yang digunakan untuk pengujian adalah komputer dengan spesifikasi yang mampu menjalankan aplikasi pendeteksi konflik pada diagram kelas.

B. Data Pengujian

Data pengujian diperoleh dengan cara penyebaran kuisioner kepada para praktisi dibidang teknologi informasi khususnya dibidang pemodelan diagram kelas. Kuisioner berupa studi khusus yang akan dibagikan kepada para responden. Kemudian para responden membuat diagram kelas dari studi kasus yang telah mereka pelajari.

C. Pengujian Kakas Bantu Pendeteksi Konflik

Diagram kelas hasil dari kuisioner kemudian diuji dengan menggunakan kakas bantu. Kemudian hasil pengujian dengan kakas bantu dibandingkan dengan hasil pengujian dengan para responden. pengujian validitas konflik dari hasil kuisioner dilakukan dengan mempertemukan dua orang responden yang diagram kelasnya terjadi konflik. Pengujian ini dilakukan dengan menggunakan metode Cohen's Kappa.

3.5 Penulisan Buku Tesis

Dalam penulisan buku tesis, bab-bab yang direncanakan untuk disusun adalah sebagai berikut:

1. Bab 1 Pendahuluan, yang berisikan latar belakang, perumusan masalah, tujuan, batasan penelitian dan kontribusi penelitian.
2. Bab 2 Tinjauan Pustaka, yang berisikan studi literatur mengenai teori-teori pengembangan model pada lingkungan banyak pengguna dan deteksi konflik.
3. Bab 3 Perencanaan Sistem, yang berisikan *flow chart*, algoritma dan desain *user interface* yang dipakai.
4. Bab 4 Implementasi Sistem, yang berisikan implementasi sistem penyelesaian deteksi konflik pada lingkungan pengembangan model dengan banyak pengguna.

5. Bab 5 Pengujian dan Analisa, yang berisikan pengujian dan analisa deteksi konflik pada lingkungan pengembangan banyak pengguna dengan menggunakan sistem yang sudah dikembangkan.
6. Bab 6 Kesimpulan dan Saran Pengembangan, yang berisikan kesimpulan dari pengujian yang dilakukan dan saran pengembangan penelitian selanjutnya.

[Halaman ini sengaja dikosongkan]

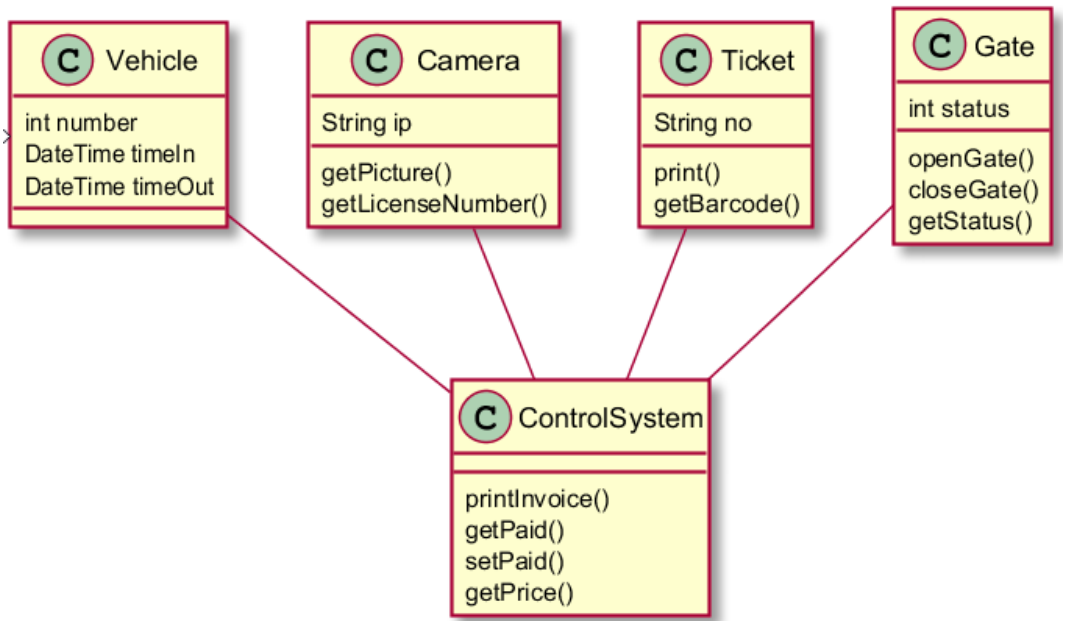
BAB 4
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Dataset

Dataset yang digunakan adalah kumpulan dari kelas diagram yang memiliki banyak versi. Pengumpulan dataset dilakukan dengan cara survei. Pengumpulan dataset dengan metode survei bertujuan untuk mendapatkan data dengan banyak versi dan memiliki tingkat konflik yang tinggi.

Pengumpulan dataset dengan metode survei dilakukan dengan menyebarkan angket yang berisi studi kasus. Studi kasus yang digunakan adalah narasi yang menjelaskan proses bisnis dari suatu sistem dan diagram kelas dari bisnis proses tersebut yang masih perlu penyempurnaan. Penyempurnaan kelas diagram dilakukan oleh masing-masing responden sehingga dari satu studi kasus menghasilkan banyak versi dan memiliki tingkat konflik yang tinggi. Responden dari survei ini adalah dosen, mahasiswa, programmer dan analis yang bekerja diperusahaan.

Studi kasus yang digunakan adalah sistem informasi parkiran mobil dengan diagram kelas pada Gambar 4.1. Diagram kelas tersebut merupakan kelas diagram yang masih membutuhkan penyempurnaan. Dalam penelitian ini diagram kelas tersebut merupakan diagram kelas versi yang ke 0 (nol). Responden diperintahkan untuk menambahkan, mengganti ataupun menghapus kelas, nama kelas, operasi dan atribut sesuai dengan pemahaman dari penjelasan studi kasus. Pengumpulan data melalui angket tersebut menghasilkan 30 versi diagram kelas. Data ini yang akan digunakan dalam proses pengujian penelitian.



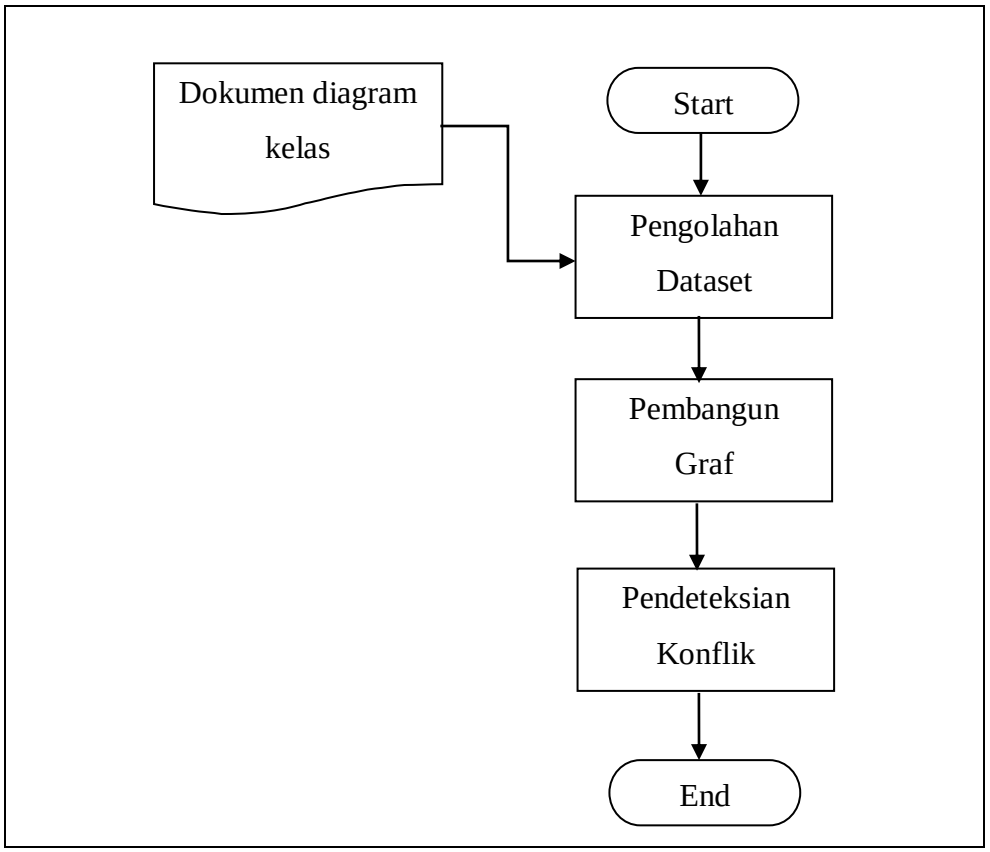
Gambar 4.1 Diagram Kelas Sistem Informasi Parkiran Mobil

4.2 Implementasi

Proses pendeteksian konflik pada versi kelas diagram dilakukan dengan dua tahap, proses otomatis dan proses manual. Proses pendeteksian secara manual dilakukan oleh pakar yang memahami tentang UML khususnya diagram kelas. Sedangkan untuk proses otomatis dilakukan dengan menggunakan kakas bantu. Kakas bantu dibangun dengan menggunakan bahasa pemrograman java dan pustaka pendukung sebagai berikut

1. WordNet *Similarity For Java* : digunakan untuk membandingkan kemiripan kata secara semantik.
2. JGraph : digunakan untuk membangun diagram kelas menjadi sebuah graf.
3. Plantuml : digunakan untuk memvisualisasikan diagram kelas.

Alur pendeteksian konflik dengan cara otomatis dapat dilihat pada Gambar 4.2 yang meliputi proses pengolahan dataset, pembangun graf dan pendeteksian konflik.



Gambar 4.2 Alur Kerja Pendeteksian Konflik Secara Otomatis

4.2.1. Pengolahan Dataset

Dataset yang akan diujikan dalam penelitian ini diperoleh dari survei. Data hasil survei yang berisi lembaran angket atau foto angket diproses terlebih dahulu agar pendeteksian konflik bisa berjalan dengan baik. Pemrosesan ini meliputi pembenahan salah ketik, pembenahan penulisan sesuai aturan konvensi, dan konversi dari angket menjadi format plantuml.

Pembenahan salah ketik dilakukan secara manual dengan melakukan pengecekan satu persatu terhadap penulisan nama komponen penyusun diagram kelas. Penulisan nama ini harus menggunakan bahasa Inggris. Jika di dalam angket yang diperoleh terdapat kata yang tidak ada dalam bahasa inggris maka perlu ditanyakan langsung ke responden maksud dari kata tersebut.

Dalam aturan konvensi bahasa pemrograman java, nama kelas, atribut dan operasi ditulis menggunakan kata kapital (*camel case*). Pembenahan diagram kelas dari responden yang tidak memenuhi aturan konvensi dilakukan secara

manual. Pembenahan ini dilakukan karena setiap frasa akan dibandingkan tingkat kesamaan maknanya.

Plantuml digunakan untuk memvisualisasikan diagram kelas mulai dari dataset hingga diagram kelas setelah pendeteksian konflik. Visualisasi ini digunakan untuk membandingkan secara manual apakah pendeteksian konflik sudah berjalan sesuai dengan hasil yang diharapkan. Diagram kelas ditulis dengan ketentuan format penulisan plantuml satu objek satu baris seperti pada Gambar 4.3 Format Penyimpanan Diagram Kelas

.

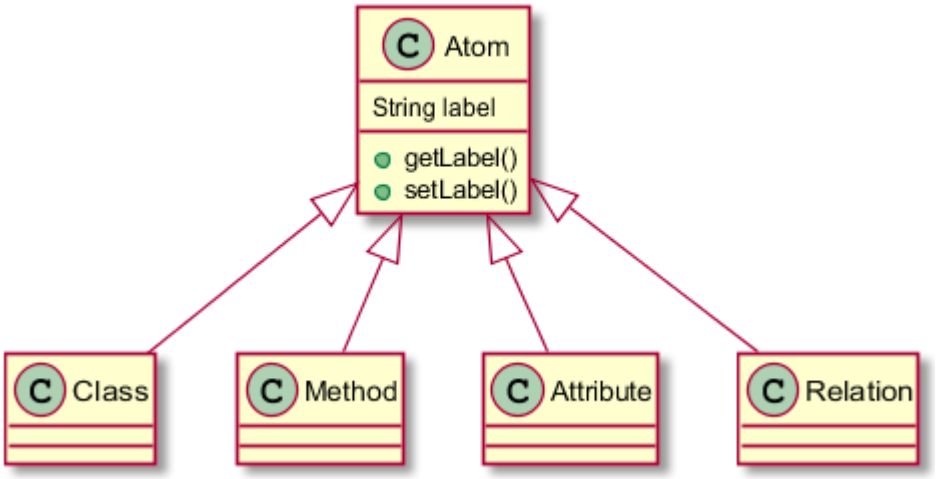
```
@startuml
Camera : String ip
Camera : getPicture()
Camera : getLicenseNumber()
ControlSystem : printInvoice()
ControlSystem : getPaid()
ControlSystem : setPaid()
ControlSystem : getPrice()
Ticket : String no
Ticket : print()
Ticket : getBarcode()
Gate : int status
Gate : openGate()
Gate : closeGate()
Gate : getStatus()
Camera -- ControlSystem
Gate -- ControlSystem
Ticket -- ControlSystem
@enduml
```

Gambar 4.3 Format Penyimpanan Diagram Kelas

Hal ini dilakukan untuk mempermudah proses pembacaan objek per baris. File dengan format plantuml ini disimpan dengan menambahkan penomoran diakhir nama file. Penomoran ini dilakukan untuk membedakan antar versi diagram kelas. Pengolahan dataset ini dilakukan untuk mempermudah sistem dalam membaca seluruh objek dalam diagram kelas secara otomatis.

4.2.2. Pembangun Graf

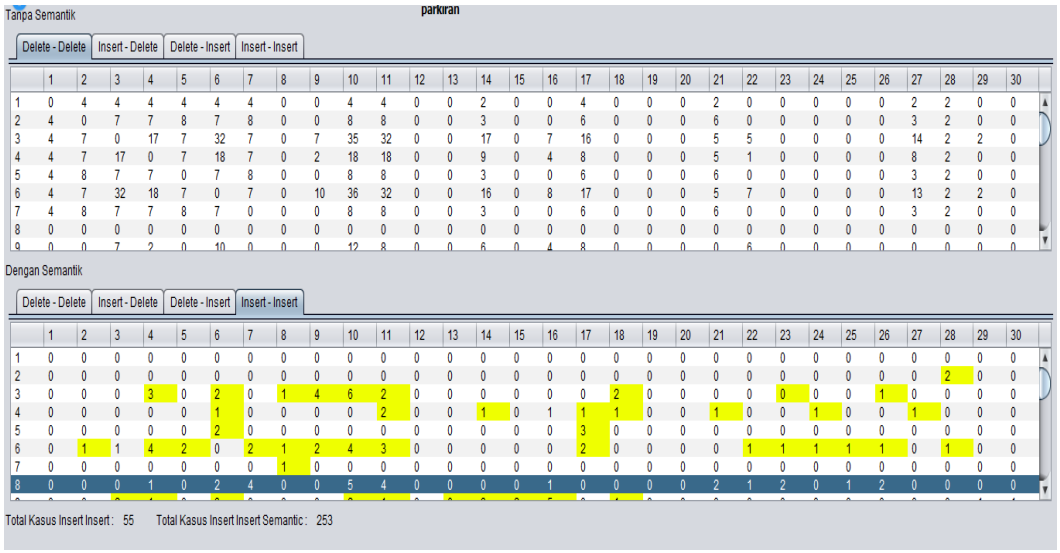
Dataset yang sudah diolah menjadi format plantuml kemudian diubah menjadi bentuk graf dengan menggunakan pustaka Jgraph. Setiap komponen dalam kelas diagram akan menjadi sebuah simpul dalam graf. Simpul dalam graf merupakan objek dari kelas atom. Komponen-komponen penyusun diagram kelas, kelas, metode, atribut dan relasi merupakan *instance* dari kelas atom. Kelas diagram komponen penyusunnya bisa dilihat pada Gambar 4.4.



Gambar 4.4 Diagram Kelas dari Objek-objek Penyusun Graf

4.2.3. Pendeteksian Konflik Dengan Graf

Proses pendeteksian konflik dilakukan pada diagram kelas yang sudah diproses menjadi bentuk graf. Pendeteksian dilakukan dengan mencari aturan minimal dari diagram kelas yang dibandingkan. Pendeteksian konflik menggunakan metode graf menghasilkan 3 buah jenis konflik *delete-delete, insert-delete, delete-insert*. Sedangkan untuk konflik *insert-insert* dengan metode modifikasi graf tidak diperhitungkan karena dalam metode modifikasi graf konflik *insert-insert* tidak dianggap sebagai konflik. Namun dalam penelitian ini konflik *insert-insert* dalam metode graf akan ditampilkan untuk bisa dibandingkan dengan metode similaritas WordNet.



Gambar 4.5 Antarmuka Hasil Deteksi Konflik

Gambar 4.5 merupakan antarmuka hasil deteksi konflik dengan metode modifikasi graf dan hasil deteksi konflk dengan similaritas WordNet.

4.2.4. Pendeteksian Konflik Dengan Similaritas WordNet

Hasil dari pendeteksian dengan metode modikasi graf adalah simpul dalam graf yang mengalami penambahan baru. Kemudian simpul baru ini akan dibandingkan kemiripannya dengan menggunakan similaritas WordNet. Proses pendeteksian konflik dengan similaritas WordNet menggunakan pustaka bantu *WordNet similarity for java (ws4j)*. Penamaan kelas, atribut dan metode mengikuti aturan penulisan sesuai konvensi. Penulisan nama pada diagram kelas menggunakan format *camel case*. Setiap kata penyusun nama dibandingkan kemiripannya menggunakan *ws4j*. Untuk memecah setiap kata yang akan dibandingkan, digunakan sebuah regex.

Setiap kata yang sudah dipecah kemudian dicari nilai similaritasnya menggunakan *ws4j*. Nilai similaritas masing-masing kata dimasukkan ke dalam matrik nilai similaritas seperti pada Tabel 2.2. Kemudian kata yang mempunyai kemiripan dipasangkan dimulai dengan nilai similaritas tertinggi ke nilai similaritas terendah. Proses pemasangan kata yang mirip ini dilakukan hingga salah satu nama memiliki kata yang sudah berpasangan semua. Kata-kata yang sudah berpasangan ini kemudian dicari rata-rata nilai similaritasnya. Nilai rata-

rata similaritas yang didapatkan dibandingkan dengan nilai ambang batas. Jika nilai rata-rata similaritas lebih tinggi dari nilai ambang batas, maka nama yang dibandingkan tersebut dapat dikatakan mirip dan terjadi konflik semantik, dan sebaliknya jika nilai rata-rata similaritas lebih rendah dari nilai ambang batas, maka nama yang dibandingkan tersebut dapat dikatakan tidak mirip dan tidak terjadi konflik.

4.3 Skenario Pengujian

Pengujian terhadap metode yang ditawarkan pada penelitian ini dilakukan dengan mengimplementasikan alur kerja yang sudah diusulkan dan dijelaskan pada metodelogi. Alur kerja tersebut diimplementasikan pada kasus data yang diangkat untuk uji coba. Data yang digunakan untuk uji coba dibagi menjadi tiga kategori yaitu ukuran kecil, sedang dan besar.

Pengujian dilakukan dengan membandingkan hasil identifikasi sistem dengan hasil dari pakar. Pakar yang dipilih untuk melakukan identifikasi adalah seorang yang paham dengan pengembangan perangkat lunak menggunakan metode pendekatan berorientasi objek dan memahami penggunaan UML terutama diagram kelas.

Proses identifikasi konflik dilakukan secara otomatis menggunakan kakas bantu. Kemudian hasil pengidentifikasian tersebut dibandingkan dengan pengidentifikasian secara manual oleh pakar. Tahap selanjutnya koefisien Kappa dihitung untuk melihat tingkat kecocokan antara hasil identifikasi kakas bantu dengan pakar.

4.3.1. Pengujian

Hasil pendeteksian yang dicatat dari proses deteksi menggunakan kerangka kerja yang diusulkan dalam penelitian ini disebut sebagai hasil deteksi sistem. Deteksi konflik dilakukan dengan dua tahap, secara otomatis dan secara manual oleh pakar. Deteksi sistem menghasilkan 991 kandidat konflik semantik. Kondidat konflik kemudian dideteksi manual oleh pakar kandidat tersebut konflik semantik atau tidak konflik. Hasil pendeteksian manual oleh pakar dapat dilihat pada Tabel 4.1

Tabel 4.1 Hasil Deteksi Konflik oleh Pakar

	Konflik	Tidak Konflik
	66	925
Total Kandidat	991	

Pada skenario pengujian dilakukan beberapa percobaan terhadap studi kasus dengan rentang ambang batas similaritas 0,1 sampai 1,0. Hasi pengujian yang dilakukan oleh sistem menghasilkan kandidat yang konflik dan tidak konflik. Hasil ini kemudian dibandingkan dengan pendeteksian konflik manual oleh pakar sehingga dapat diketahui kandidat konflik yang salah dalam pendeteksian. Perbandingan hasil deteksi sistem dengan hasil deteksi pakar dapat dilihat pada Tabel 4.2.

Tabel 4.2 Perbandingan Hasil Deteksi Sistem

Ambang Batas Similaritas	Konflik	Tidak Konflik	Salah deteksi
0,1	991	0	925
0,2	991	0	925
0,3	829	162	763
0,4	656	335	590
0,5	526	465	460
0,6	253	738	189
0,7	127	864	89
0,8	49	942	39
0,9	32	959	38
1,0	27	964	43

4.3.2. Analisis hasil pengujian

Dari Tabel 4.2, dapat dijelaskan bahwa pendeteksian konflik dengan jumlah kesalahan deteksi tertinggi yaitu sebesar 925. Kesalahan deteksi tertinggi ini terdapat pada ambang batas similaritas 0,1 dan 0,2. Tingginya jumlah kesalahan deteksi disebabkan karena pada ambang batas similaritas 0,1 dan 0,2

semua kandidat konflik terdeteksi sebagai kasus konflik semantik. Sedangkan pendeteksian dengan tingkat kesalah terendah terdapat pada ambang batas similaritas 0,9 dengan jumlah kesalahan deteksi sebesar 38.

Dari kedua hasil pada Tabel 4.1 dan Tabel 4.2, maka koefisien Kappa bisa dihitung dengan menggunakan persamaan 8. Hasil perhitungan koefisien kappa dengan ambang batas similaritas antara 0,1 dan 1,0 dapat dilihat pada Tabel 4.3 Perhitungan Koefisien Kappa.

Tabel 4.3 Perhitungan Koefisien Kappa

Ambang Batas Similaritas	Koefisien Kappa
0,1	0,0000
0,2	0,0000
0,3	0.0275
0,4	0.0703
0,5	0.1186
0,6	0.3375
0,7	0,4945
0,8	**0,6404**
0,9	0,5946
1,0	0,5190

Dari tabel diatas, dapat dijelaskan bahwa nilai kappa terendah terdapat pada ambang batas 0,1 dan 0,2. Sedangkan nilai kappa tertinggi terdapat pada ambang batas 0,8 yaitu 0,6404. Dari hasil koefisien Kappa tersebut maka dapat diinterpretasikan bahwa hasil tingkat kecocokan antara sistem dan pakar pada penelitian ini adalah kuat (*good*).

[Halaman ini sengaja dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

Bab ini menjelaskan tentang apa saja yang dapat disimpulkan dari penelitian yang sudah dilakukan. Kemudian, peneliti menjelaskan bagaimana saran atau kelanjutan penelitian pada masa mendatang.

5.1 Kesimpulan

Beberapa kesimpulan yang dapat ditarik dari hasil pengerjaan penelitian ini adalah sebagai berikut.

1. Dalam penelitian ini dikembangkan suatu metode untuk mendeteksi konflik secara leksikal pada diagram kelas. Metode yang digunakan adalah modifikasi graf dan similaritas WordNet.
2. Hasil akhir uji coba oleh sistem dibandingkan dengan uji coba oleh pakar. Pendekatan ini bertujuan untuk mendapatkan koefisien Kappa dari metode yang diusulkan dalam penelitian ini. Nilai kappa tertinggi sebesar 0,6404 diperoleh ketika menggunakan nilai ambang batas similaritas 0,8.
3. Hasil perhitungan koefisien Kappa adalah 0,6404 yang kemudian dapat diinterpretasikan bahwa hasil tingkat kecocokan antara sistem dan pakar pada penelitian ini adalah kuat (*good*).
4. Dalam penelitan ini menggunakan studi kasus diagram kelas sistem informasi parkiran mobil. Nilai kappa yang diperoleh belum tentu sama ketika diujikan pada studi kasus yang lain.

5.2 Saran

Saran untuk penelitian selanjutnya adalah sebagai berikut.

1. Penelitian ini menggunakan dataset diagram kelas yang diperoleh dari menyebar angket. Hasil diagram kelas dari masing-masing responden mewakili perbedaan versi dalam kontrol versi. Pada penelitian mendatang perlu dikembangkan lingkungan kontrol versi diagram kelas yang sudah

dilengkapi dengan kakas pembuat diagram kelas dan pendeteksi konflik baik konflik secara sintaksis maupun konflik secara semantik.

2. Penelitian ini hanya menggunakan satu studi kasus yaitu diagram kelas sistem informasi parkiran mobil. Nilai koefisian kappa yang diperoleh belum tentu sama jika diterapkan pada kasus yang lain. Oleh karena itu meningkatkan keakuratan nilai kappa perlu diuji cobakan pada studi kasus yang lain.
3. Dalam penelitian ini nilai similaritas dihitung dengan membandingkan similaritas nama kelas, nama atribut dan nama operasi, tipe data dalam atribut dan operasi tidak diperhitungkan. Dalam sebuah kelas diperbolehkan menulis operasi dengan nama yang sama tetapi dengan tipe data berbeda. Oleh karena itu dalam penelitian yang akan datang, perlu diperhitungkan juga similaritas tipe data.

DAFTAR PUSTAKA

Brosch, Petra., Kargl, Horst., Seidl, Martina., Wieland, Konrad., Wimmer, Manuel., dan Kappel, Gerti.,(2011), "*Conflict as First-Class Entities: A UML Profile for Model Versioning*", MODELS 2010 Workshop, Springer-Verlag, Berlin Heidelberg.

Miller, George A., Beckwith, Richard., Fellbaum, Christiane., Gross, Derek., dan Miller, Katherine., (1993), "*Introduction to WordNet: An On-line Lexical Database*"

Altmanninger, K., dan Pierantonio, A., (2011), "*A categorization for conflicts in model versioning*", Elektrotechnik & Informationstechnik, Springer

Ehrig, Hartmut., Ermel, Claudia., dan Taentzer, Gabriele. (2011), "*A Formal Resolution Strategy for Operation-Based Conflicts in Model Versioning Using Graph Modifications*".

Rajbhoj, Asha., dan Reddy, Sreedhar., (2013), "*A Graph-Pattern Based Approach for Meta-Model Specific Conflict Detection in a General-Purpose Model Versioning System*"

Taentzer, Gabriele., Ermel, Claudia., Langer, Philip., dan Wimmer., Manuel, (2010), "*Conflict Detection for Model Versioning Based on Graph Modifications*"

Gomes, Paulo., C Pereira, Francisco., Paiva, Paulo., Seco, Nuno., Carreiro, Paulo., L. Ferreira, José., dan Bento, Carlos., (2000), "*Case Retrieval of Software Designs using WordNet*"

Brosch, Petra., Seidl, Martina., Wieland, Konrad., dan Wimmer, Manuel., (2009), "*We can work it out: Collaborative Conflict Resolution in Model Versioning*"

Altman, DG, (1991), "*Practical Statistics for Medical Research*"

Sim, J., & Wright, C. C., (2005), "The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements", *Physical Therapy*, Vol. 85(3), hal. 257–68.

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS

Penulis dilahirkan di Lumajang pada tanggal 28 Agustus 1987, yang merupakan anak kedua dari dua bersaudara. Penulis telah menempuh pendidikan dasar di SDN Yosowilangun Lor 1, SLTP Negeri 1 Yosowilangun, dan SMA Negeri 2 Lumajang. Pada tahun 2005 penulis melanjutkan pendidikan ke jenjang pendidikan tinggi S1 Program Studi Teknik Informatika di Institut Teknologi Sepuluh Nopember dan lulus tahun 2009. Setelah itu, penulis bekerja sebagai programmer diperusahaan swasta. Kemudian, tahun 2011 penulis melanjutkan studi S2 di Institut Teknologi Sepuluh Nopemeber Program Studi Teknik Informatika. Penulis mengambil bidang minat Rekayasa Perangkat Lunak baik di jenjang S1 maupun S2. Untuk korespondensi, penulis dapat dihubungi melalui email billy.montolalu@gmail.com.